MATLAB
开发实例系列图书

MATLAB 神经网络 30个案例分析

MATLAB中文论坛 编著

北京航空航天大学出版社

MATLAB 神经网络 30 个案例分析

MATLAB 中文论坛　编著

北京航空航天大学出版社

内 容 简 介

本书是 MATLAB 中文论坛神经网络版块数千个帖子的总结，充分强调“案例实用性、程序可模仿性”。所有案例均来自于论坛会员的切身需求，保证每一个案例都与实际课题相结合。

读者调用案例的时候，只要把案例中的数据换成自己需要处理的数据，即可实现自己想要的网络。如果在实现过程中有任何疑问，可以随时在 MATLAB 中文论坛与作者交流，作者每天在线，有问必答。

该书共有 30 个 MATLAB 神经网络的案例（含可运行程序），包括 BP、RBF、SVM、SOM、Hopfield、LVQ、Elman、小波等神经网络；还包含 PSO（粒子群）、灰色神经网络、模糊网络、概率神经网络、遗传算法优化等内容。该书另有 31 个配套的教学视频帮助读者更深入地了解神经网络。

本书可作为本科毕业设计、研究生项目设计、博士低年级课题设计参考书籍，同时对广大科研人员也有很高的参考价值。

图书在版编目(CIP)数据

MATLAB 神经网络 30 个案例分析/MATLAB 中文论坛编著. --北京：北京航空航天大学出版社，2010.4

ISBN 978-7-5124-0034-4

Ⅰ. ①M… Ⅱ. ①M… Ⅲ. ①计算机辅助计算—软件包，Matlab—应用—神经网络—案例—分析 Ⅳ. ①TP391.75②TP183

中国版本图书馆 CIP 数据核字(2010)第 039648 号

MATLAB 神经网络 30 个案例分析

MATLAB 中文论坛 编著

责任编辑 陈守平 董 瑞 罗晓莉

*

北京航空航天大学出版社出版发行

北京市海淀区学院路 37 号(100191) 发行部电话：(010)82317024 传真：(010)82328026

http://www.buaapress.com.cn E-mail:bhpress@263.net

印刷有限公司印装 各地书店经销

*

开本：787×1 092 1/16 印张：18.5 字数：474 千字

2010 年 4 月第 1 版 2010 年 4 月第 1 次印刷 印数：4 000 册

ISBN 978-7-5124-0034-4 定价：39.00 元

前　言

MATLAB 中文论坛神经网络版块有数千个 MATLAB 与神经网络相关的帖子。我们对这些帖子进行了一些总结分析，发现一些比较有趣的现象：

① 大约有 20％的会员不知道每一种神经网络的功能，不清楚该选用何种神经网络来做自己的课题。

② 大约有 50％的会员会直接参考他人已经写好的代码，然而由于数据性质、应用背景等的差异性，会员在修改现有代码使之更符合自己的需要时遇到很多麻烦。

③ 还有一小部分会员想了解如何让现有的神经网络与其他方面的优化知识结合起来，使神经网络的表现更理想一些，比如神经网络与遗传算法的结合等，但在现有很多有关神经网络的书上找不到答案。

在我们回答问题的同时，我们对现有的提问进行了分析和总结，尤其是会员比较关心的以上现象进行了统计。为了让更多学习神经网络的会员能够快速了解并且在MATLAB下使用神经网络，MATLAB 中文论坛精心编写了《MATLAB 神经网络 30 个案例分析》一书。

该书含有 30 个在 MATLAB 环境下实现的神经网络案例，包括了常用的神经网络及相关理论，如：BP、RBF、SVM、SOM、PSO、Hopfield、Elman、LVQ、Kohonen、GRNN、灰色神经网络、遗传算法与神经网络的结合、广义神经网络、小波神经网络、PID 神经元等知识。当然，如果你所需要的神经网络超出本书所涉及的范畴，收到你的反馈后几位作者会第一时间在论坛“在线交流”版块为你加上。别忘了，这是一本“会动”的书！

在编写本书的过程中，我们始终记得数千位会员对该书的要求（希望这也是你的期待）：

案例实用性。书中所列举的 30 个案例，部分来自于各大公司、院校的研究课题，部分来自于论坛会员的提问。这些案例分别代表了神经网络在各个领域的相关应用。读者可以很容易根据自己的课题需要，找出书中哪些案例适合自己，进而详细阅读。

MATLAB 程序可模仿性。我们所编写的 MATLAB 案例程序高度模块化。不管是何种网络，其基本思想都是输入（出）数据的前期处理、模型参数的设置、模型的训练以及模型的使用。那么，如果读者需要模仿这些程序，只需更改里面某些模块即可。

我们一直强调一个理念：“有问必答”！对于神经网络这门学科来说，包含太多的抽象知识。如果在学习、使用神经网络的时候，能够得到一位或者数位该领域专家的指导，那绝不仅仅是事半功倍的效果，我想学习过编程语言的读者都知道这个道理。目前这本书的几位作者几乎每天都在线为读者解答疑问，争取做到问题不过夜。

对于每个案例，我们也制作了配套的教学视频。在书籍＋视频＋程序的协助下，**一小时之内使用神经网络实现自己的目标已经不是难事。**套一句比较流行的网络术语：“哥卖的不是书，而是一种服务”。

我们特别想感谢“你”，因为你经常在 MATLAB 中文论坛为此书出谋划策，因为你细心地阅读每个案例、程序，然后指出需要改进的地方，才能让此书符合大部分会员的需求。（注：此书采取开放式编著方式，MATLAB 中文论坛对此书提供及时的报道，如：书籍的写作进度、书

籍的程序案例等，所以很多读者是看着这本书如何“诞生”的。）

该书适合所有做与神经网络相关研究的读者阅读，当然我们的特长是在 MATLAB 下面实现神经网络的各种应用。

史峰（网名 shi01fg，超级版主）编写了第 1～ 6、23～26、28、29 章；王小川（网名 hgsz2003，超级版主）编写了第 7、8、16～20、30 章；郁磊（网名 yuthreestone，超级版主）编写了第 9 ～ 11、21 、22 、27 章； 李洋（网名 faruto，超级版主）编写了第 12～15 章；张延亮（网名 math，论坛站长）负责全部案例的选取和审核；焦小雪（网名 mooni，论坛管理员）负责所有配套视频下载的测试与维护。

我们深知，神经网络就是一个巨大的黑盒子，我们一直在研究其精华奥妙所在，如果在我们的探索路上有任何不当之处，希望读者能及时向我们反馈。请你相信，我们的纠错速度也会超乎你的想象！

最后，给大家一个小小的承诺：如果你在阅读完该书的一个案例，或者看完一个视频以后，还是不懂得如何在 MATLAB 下实现神经网络，你可以在 MATLAB 中文论坛该书“在线交流”版块发帖，论坛会安排一位本书的作者亲自指导你，直到你得到自己想要的程序为止！

MATLAB 中文论坛
2009 年 12 月于南京

该书作者简介

史峰

史峰，网名 shi01fg，MATLAB 中文论坛超级版主。在 MATLAB 中文论坛长期解答各种 MATLAB 疑难问题。从 MATLAB 6.5 开始接触 MATLAB 软件，主要用于科学计算和工程开发，并有多个 MATLAB 工程开发经验。在长期的使用过程中积累了较丰富的编程经验，擅长于神经网络、智能算法、科学计算和 GUI 设计。

王小川

王小川，网名 hgsz2003，MATLAB 中文论坛超级版主。现于哈尔滨医科大学公共卫生学院卫生统计学攻读硕士学位，长期研究神经网络在统计学中的应用，精通 MATLAB、SAS、SPSS 等统计软件，热衷数据分析和数据挖掘工作，擅长竞争神经网络在数据挖掘中的运用，有着扎实的理论基础和丰富的实战经验。

郁磊

郁磊，网名 yuthreestone，MATLAB 中文论坛超级版主。现于中国矿业大学信息与电气工程学院攻读控制理论与控制工程专业硕士学位。研究方向为智能控制与模式识别，对神经网络、遗传算法、支持向量机等智能算法有深入的研究。使用 MATLAB 六年多来，对 MATLAB 环境及工具箱函数非常熟悉，具有丰富的 MATLAB 开发、设计经验。

李洋

李洋，网名 faruto，MATLAB 中文论坛超级版主。现于北京师范大学数学科学学院攻读硕士学位，专业为应用数学，主要方向为模糊数学及人工智能。目前关注领域为信息粒化在时间序列和机器学习中的应用，SVM 相应参数的优化问题以及 SVM 的应用问题。对 SVM 工具箱的使用有丰富的经验，对现有的 SVM 工具箱进行了大量的优化提升等工作，同时擅长优化理论等。

张延亮

张延亮，网名 math，MATLAB 中文论坛站长，百度 MATLAB 专家。2006 年创建 MATLAB 中文论坛，组织了近 10 次大型的 MATLAB/Simulink 研讨会。长期在论坛里解答会员问题。研究方向包含图像处理、人工智能、高级控制理论、生物信息系统等。负责“在线交流，有问必答”部分书籍的选题、写作与审核工作，确保相关书籍能真正填补市场空白，满足读者需求。

目　　录

络。要求该网络可以正确地识别 0～9 这 10 个数字，当数字被一定的噪声干扰后，仍具有较好的识别效果。

第 1 章　BP 神经网络的数据分类

——语音特征信号分类

1.1　案例背景

1.1.1　BP 神经网络概述

BP 神经网络是一种多层前馈神经网络，该网络的主要特点是信号前向传递，误差反向传播。在前向传递中，输入信号从输入层经隐含层逐层处理，直至输出层。每一层的神经元状态只影响下一层神经元状态。如果输出层得不到期望输出，则转入反向传播，根据预测误差调整网络权值和阈值，从而使 BP 神经网络预测输出不断逼近期望输出。BP 神经网络的拓扑结构如图 1－1 所示。

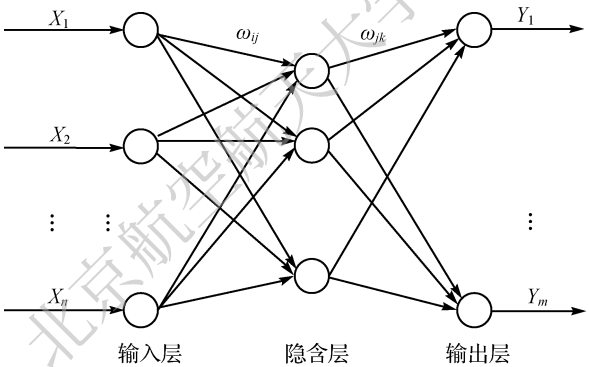

图 1－1　BP 神经网络拓扑结构图

图 1－1 中，$X_1, X_2, \cdots, X_n$ 是 BP 神经网络的输入值，$Y_1, Y_2, \cdots, Y_m$ 是 BP 神经网络的预测值，ω_{ij} 和 ω_{jk} 为 BP 神经网络权值。从图 1－1 可以看出，BP 神经网络可以看成一个非线性函数，网络输入值和预测值分别为该函数的自变量和因变量。当输入节点数为 n，输出节点数为 m 时，BP 神经网络就表达了从 n 个自变量到 m 个因变量的函数映射关系。

BP 神经网络预测前首先要训练网络，通过训练使网络具有联想记忆和预测能力。BP 神经网络的训练过程包括以下几个步骤。

步骤 1：网络初始化。根据系统输入输出序列(X,Y)确定网络输入层节点数 n、隐含层节点数 l，输出层节点数 m，初始化输入层、隐含层和输出层神经元之间的连接权值 ω_{ij}，ω_{jk}，初始化隐含层阈值 a，输出层阈值 b，给定学习速率和神经元激励函数。

步骤 2：隐含层输出计算。根据输入向量 X，输入层和隐含层间连接权值 ω_{ij} 以及隐含层阈值 a，计算隐含层输出 H。

$$H_j = f\left(\sum_{i=1}^{n}\omega_{ij}x_i - a_j\right) \quad j = 1,2,\cdots,l \tag{1-1}$$

式中，l 为隐含层节点数；f 为隐含层激励函数，该函数有多种表达形式，本章所选函数为：

$$f(x) = \frac{1}{1+e^{-x}} \tag{1-2}$$

步骤 3：输出层输出计算。根据隐含层输出 H，连接权值 ω_{jk} 和阈值 b，计算 BP 神经网络预测输出 O。

$$O_k = \sum_{j=1}^{l}H_j\omega_{jk} - b_k \quad k = 1,2,\cdots,m \tag{1-3}$$

步骤 4：误差计算。根据网络预测输出 O 和期望输出 Y，计算网络预测误差 e。

$$e_k = Y_k - O_k \quad k = 1,2,\cdots,m \tag{1-4}$$

步骤 5：权值更新。根据网络预测误差 e 更新网络连接权值 ω_{ij}，ω_{jk}。

$$\omega_{ij} = \omega_{ij} + \eta H_j(1-H_j)x(i)\sum_{k=1}^{m}\omega_{jk}e_k \quad j = 1,2,\cdots,n;j = 1,2,\cdots,l \tag{1-5}$$

$$\omega_{jk} = \omega_{jk} + \eta H_j e_k \quad j = 1,2,\cdots,l;k = 1,2,\cdots,m \tag{1-6}$$

式中，η 为学习速率。

步骤 6：阈值更新。根据网络预测误差 e 更新网络节点阈值 a，b。

$$a_j = a_j + \eta H_j(1-H_j)\sum_{k=1}^{m}\omega_{jk}e_k \quad j = 1,2,\cdots,l \tag{1-7}$$

$$b_k = b_k + e_k \quad k = 1,2,\cdots,m \tag{1-8}$$

步骤 7：判断算法迭代是否结束，若没有结束，返回步骤 2。

1.1.2 语音特征信号识别

语音特征信号识别是语音识别研究领域中的一个重要方面，一般采用模式匹配的原理解决。语音识别的运算过程为：首先，待识别语音转化为电信号后输入识别系统，经过预处理后用数学方法提取语音特征信号，提取出的语音特征信号可以看成该段语音的模式。然后将该段语音模型同已知参考模式相比较，获得最佳匹配的参考模式为该段语音的识别结果。语音识别流程如图 1-2 所示。

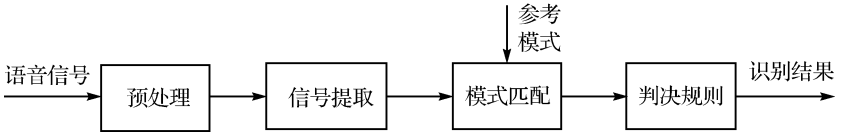

图 1-2　语音识别流程

本案例选取了民歌、古筝、摇滚和流行四类不同音乐，用 BP 神经网络实现对这四类音乐的有效分类。每段音乐都用倒谱系数法提取 500 组 24 维语音特征信号，提取出的语音特征信号如图 1-3 所示。

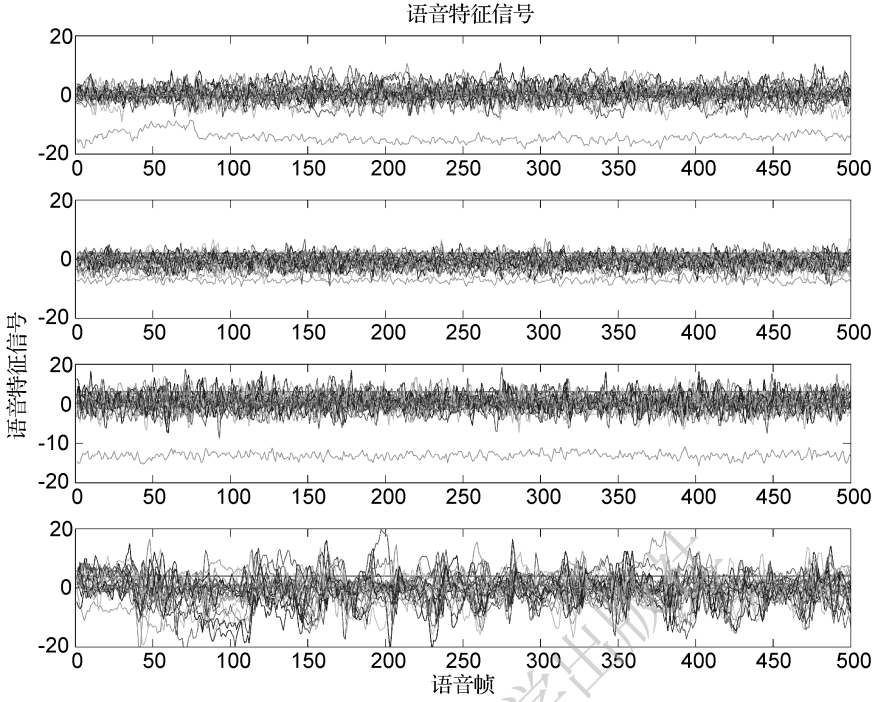

图 1-3　语音特征信号

1.2　模型建立

基于 BP 神经网络的语音特征信号分类算法建模包括 BP 神经网络构建、BP 神经网络训练和 BP 神经网络分类三步，算法流程如图 1-4 所示。

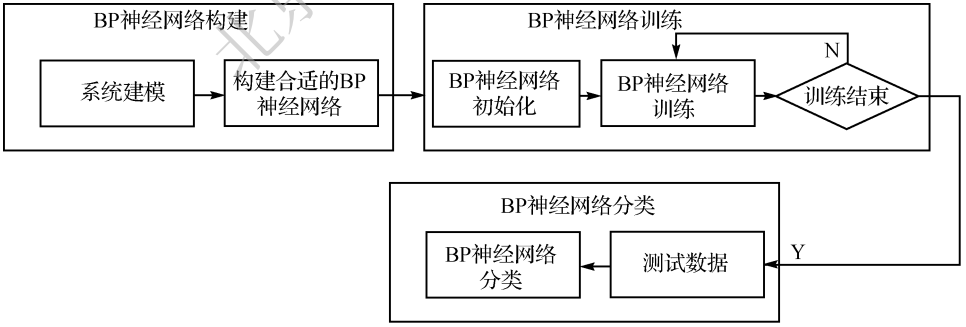

图 1-4　算法流程

BP 神经网络构建根据系统输入输出数据特点确定 BP 神经网络的结构，由于语音特征输入信号有 24 维，待分类的语音信号共有 4 类，所以 BP 神经网络的结构为 24—25—4，即输入层有 24 个节点，隐含层有 25 个节点，输出层有 4 个节点。

BP 神经网络训练用训练数据训练 BP 神经网络。共有 2 000 组语音特征信号，从中随机选择 1 500 组数据作为训练数据训练网络，500 组数据作为测试数据测试网络分类能力。

BP 神经网络分类用训练好的神经网络对测试数据所属语音类别进行分类。

1.3 MATLAB 实现

根据 BP 神经网络理论，在 MATLAB 软件中编程实现基于 BP 神经网络的语音特征信号分类算法。

1.3.1 归一化方法及 MATLAB 函数

数据归一化方法是神经网络预测前对数据常做的一种处理方法。数据归一化处理把所有数据都转化为[0,1]之间的数，其目的是取消各维数据间数量级差别，避免因为输入输出数据数量级差别较大而造成网络预测误差较大。数据归一化的方法主要有以下两种。

1）最大最小法。函数形式如下：

$$x_k = (x_k - x_{\min})/(x_{\max} - x_{\min}) \tag{1-9}$$

式中，$x_{\min}$为数据序列中的最小数；$x_{\max}$为序列中的最大数。

2）平均数方差法，函数形式如下：

$$x_k = (x_k - x_{\text{mean}})/x_{\text{var}} \tag{1-10}$$

式中，x_{mean}为数据序列的均值；x_{var}为数据的方差。

本案例采用第一种数据归一化方法，归一化函数采用 MATLAB 自带函数 mapminmax，该函数有多种形式，常用的方法如下。

```
% input_train,output_train 分别是训练输入、输出数据
[inputn,inputps] = mapminmax(input_train);
[outputn,outputps] = mapminmax(output_train);
```

input_train，output_train 是训练输入、输出原始数据，inputn，outputn 是归一化后的数据，inputps，outputps 为数据归一化后得到的结构体，里面包含了数据最大值、最小值和平均值等信息，可用于测试数据归一化和反归一化。测试数据归一化和反归一化程序如下。

```
inputn_test = mapminmax('apply',input_test,inputps);    % 测试输入数据归一化
BPoutput = mapminmax('reverse',an,outputps);            % 网络预测数据反归一化
```

input_test 是预测输入数据，inputn_test 是归一化后的预测数据，'apply'表示根据 inputps 的值对 input_test 进行归一化。an 是网络预测结果，outputps 是训练输出数据归一化得到的结构体，BPoutput 是反归一化之后的网络预测输出，'reverse'表示对数据进行反归一化。

1.3.2 数据选择和归一化

首先根据倒谱系数法提取四类音乐语音特征信号，不同的语音信号分别用 1，2，3，4 标识，提取出的信号分别存储于 data1. mat，data2. mat，data3. mat，data4. mat 数据库文件中，每组数据为 25 维，第 1 维为类别标识，后 24 维为语音特征信号。把四类语音特征信号合为一组，从中随机选取 1 500 组数据作为训练数据，500 组数据作为测试数据，并对训练数据进行归一化处理。根据语音类别标识设定每组语音信号的期望输出值，如标识类为 1 时，期望输出向量为[1　0　0　0]。

```
%清空环境变量
clc
clear

%导入四类语音信号
load data1 c1
load data2 c2
load data3 c3
load data4 c4

%将四类语音特征信号合并为一组
data(1:500,:) = c1(1:500,:);
data(501:1000,:) = c2(1:500,:);
data(1001:1500,:) = c3(1:500,:);
data(1501:2000,:) = c4(1:500,:);

%输入输出数据
input = data(:,2:25);
output1  = data(:,1);

%设定每组输入输出信号
for i = 1:2000
    switch output1(i)
        case 1
            output(i,:) = [1 0 0 0];
        case 2
            output(i,:) = [0 1 0 0];
        case 3
            output(i,:) = [0 0 1 0];
        case 4
            output(i,:) = [0 0 0 1];
    end
end

%从中随机抽取 1500 组数据作为训练数据,500 组数据作为预测数据
k = rand(1,2000);
[m,n] = sort(k);

input_train = input(n(1:1500),:)';
output_train = output(n(1:1500),:)';
input_test = input(n(1501:2000),:)';
output_test = output(n(1501:2000),:)';

%输入数据归一化
[inputn,inputps] = mapminmax(input_train);
```

1.3.3 BP 神经网络结构初始化

根据语音特征信号特点确定 BP 神经网络的结构为 24—25—4，随机初始化 BP 神经网络权值和阈值。

```
%网络结构
innum = 24;
midnum = 25;
outnum = 4;

%权值阈值初始化
w1 = rands(midnum,innum);
b1 = rands(midnum,1);
w2 = rands(midnum,outnum);
b2 = rands(outnum,1);
```

1.3.4 BP 神经网络训练

用训练数据训练 BP 神经网络，在训练过程中根据网络预测误差调整网络的权值和阈值。

```
for ii = 1:20
    E(ii) = 0;   %训练误差
    for i = 1:1:1500

        %选择本次训练数据
        x = inputn(:,i);

        %隐含层输出
        for j = 1:1:midnum
            I(j) = inputn(:,i)' * w1(j,:)' + b1(j);
            Iout(j) = 1/(1 + exp( - I(j)));
        end
        %输出层输出
        yn = w2' * Iout' + b2;

        %预测误差
        e = output_train(:,i) - yn;
        E(ii) = E(ii) + sum(abs(e));

        %计算 w2,b2 调整量
        dw2 = e * Iout;
        db2 = e';

        %计算 w1,b1 调整量
        for j = 1:1:midnum
            S = 1/(1 + exp( - I(j)));
```

```
            FI(j) = S * (1 - S);
        end
        for k = 1:1:innum
            for j = 1:1:midnum
              dw1(k,j) = FI(j) * x(k) * (e(1) * w2(j,1) + e(2) * w2(j,2) + e(3) * w2(j,3) + e(4) * w2(j,4));
              db1(j) = FI(j) * (e(1) * w2(j,1) + e(2) * w2(j,2) + e(3) * w2(j,3) + e(4) * w2(j,4));
            end
        end

        % 权值阈值更新
        w1 = w1_1 + xite * dw1';
        b1 = b1_1 + xite * db1';
        w2 = w2_1 + xite * dw2';
        b2 = b2_1 + xite * db2';

        % 结果保存
        w1_1 = w1;
        w2_1 = w2;
        b1_1 = b1;
       b2_1 = b2;
    end
end
```

1.3.5 BP 神经网络分类

用训练好的 BP 神经网络分类语音特征信号,根据分类结果分析 BP 神经网络分类能力。

```
% 输入数据归一化
inputn_test = mapminmax('apply',input_test,inputps);

% 网络预测
for i = 1:500
    for j = 1:1:midnum
        I(j) = inputn_test(:,i)' * w1(j,:)' + b1(j);
        Iout(j) = 1/(1 + exp( - I(j)));
    end
    % 预测结果
    fore(:,i) = w2' * Iout' + b2;
end

% 类别统计
for i = 1:500
    output_fore(i) = find(fore(:,i) == max(fore(:,i)));
end
% 预测误差
error = output_fore - output1(n(1501:2000))';
```

```
k = zeros(1,4);
%统计误差
for i = 1:500
    if error(i)~ = 0
        [b,c] = max(output_test(:,i));
        switch c
            case 1
                k(1) = k(1) + 1;
            case 2
                k(2) = k(2) + 1;
            case 3
                k(3) = k(3) + 1;
            case 4
                k(4) = k(4) + 1;
        end
    end
end

%统计正确率
rightridio = (kk - k)./kk
```

1.3.6 结果分析

用训练好的 BP 神经网络分类语音特征信号测试数据，BP 神经网络分类误差如图 1-5 所示。

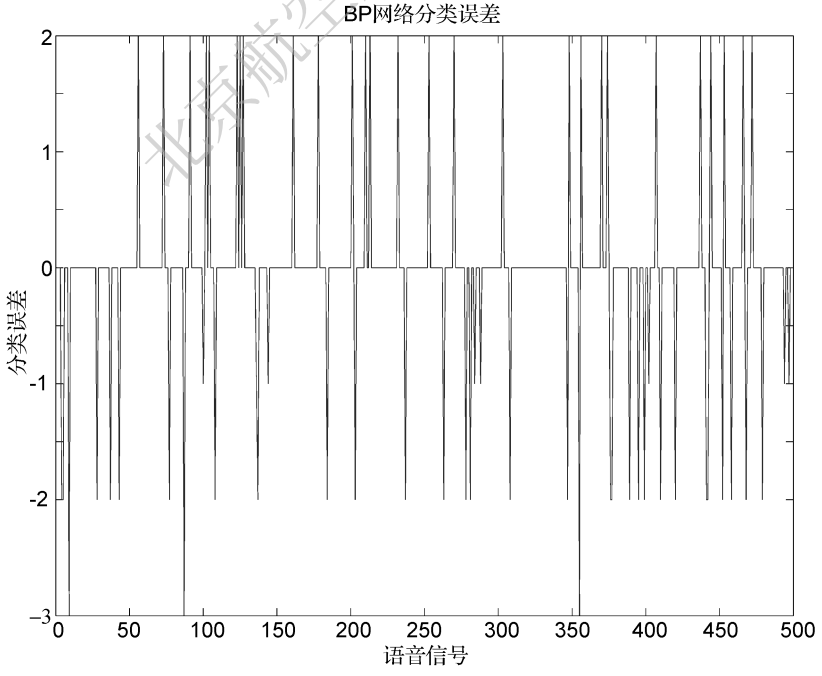

图 1-5 BP 神经网络分类误差

BP 神经网络分类正确率如表 1－1 所列。

表 1－1　BP 网络分类正确率

语音信号类别	第一类	第二类	第三类	第四类
识别正确率	0.729 3	1.000 0	0.877 0	0.956 9

从 BP 神经网络分类结果可以看出，基于 BP 神经网络的语音信号分类算法具有较高的准确性，能够准确识别出语音信号所属类别。

1.4　案例扩展

1.4.1　隐含层节点数

BP 神经网络的隐含层节点数对 BP 神经网络预测精度有较大的影响：节点数太少，网络不能很好地学习，需要增加训练次数，训练的精度也受影响；节点数太多，训练时间增加，网络容易过拟合。最佳隐含层节点数选择可参考如下公式：

$$l < n-1 \tag{1-11}$$

$$l < \sqrt{(m+n)}+a \tag{1-12}$$

$$l = \log_2 n \tag{1-13}$$

式中，n 为输入层节点数；l 为隐含层节点数；m 为输出层节点数；a 为 0～10 之间的常数。在实际问题中，隐含层节点数的选择首先是参考公式来确定节点数的大概范围，然后用试凑法确定最佳的节点数。对于某些问题来说，隐含层节点数对输出结果影响较小，如对于本案例来说，分类误差同隐含层节点数的关系如图 1－6 所示。

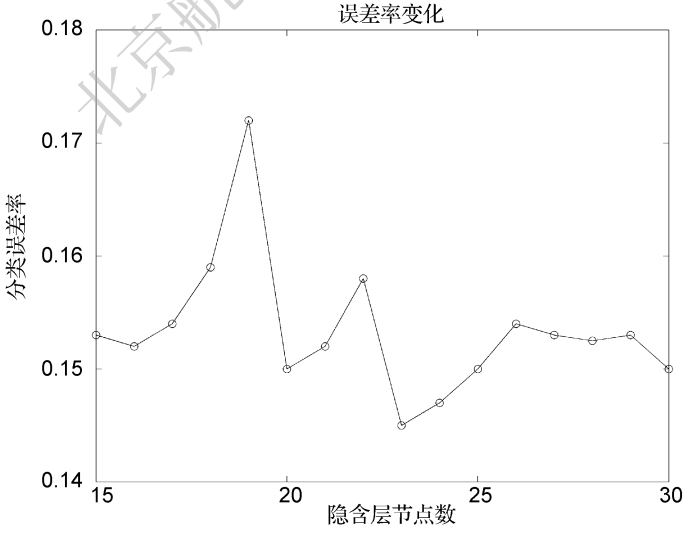

图 1－6　预测误差和隐含层节点数关系

从图 1－6 可以看出，本案例中 BP 神经网络分类误差率随着隐含层节点数的增加而减小。对

于一般问题来说，BP 神经网络的分类误差随着隐含层节点数的增加呈现先减少后增加的趋势。

1.4.2 附加动量方法

BP 神经网络的采用梯度修正法作为权值和阈值的学习算法，从网络预测误差的负梯度方向修正权值和阈值，没有考虑以前经验的积累，学习过程收敛缓慢。对于这个问题，可以采用附加动量方法来解决，带附加动量的权值学习公式为

$$\omega(k) = \omega(k-1) + \Delta\omega(k) + a[\omega(k-1) - \omega(k-2)] \tag{1-14}$$

式中，$\omega(k)$，$\omega(k-1)$，$\omega(k-2)$分别时 k，$k-1$，$k-2$ 时刻的权值；a 为动量学习率。MATLAB 程序如下：

```
% xite,alfa 为学习率
w1 = w1_1 + xite * dw1' + alfa * (w1_1 - w1_2);
b1 = b1_1 + xite * db1' + alfa * (b1_1 - b1_2);
w2 = w2_1 + xite * dw2' + alfa * (w2_1 - w2_2);
b2 = b2_1 + xite * db2' + alfa * (b2_1 - b2_2);
```

1.4.3 变学习率学习算法

BP 神经网络学习率 η 的取值在[0，1]之间，学习率 η 越大，对权值的修改越大，网络学习速度越快。但过大的学习速率 η 将使权值学习过程中产生震荡，过小的学习概率使网络收敛过慢，权值难以趋于稳定。变学习率方法是指学习概率 η 在 BP 神经网络进化初期较大，网络收敛迅速，随着学习过程的进行，学习率不断减小，网络趋于稳定。变学习率计算公式为

$$\eta(t) = \eta_{max} - t(\eta_{max} - \eta_{min})/t_{max} \tag{1-15}$$

式中，η_{max} 为最大学习率；η_{min} 为最小学习率；t_{max} 为最大迭代次数；t 为当前迭代次数。

参考文献

[1] 李丽霞. BP 神经网络及其在疾病预后分类问题中的应用[D]. 太原：山西医科大学，2002.

[2] 孟治国. BP 神经网络在土地利用分类中的应用分析[D]. 长春：吉林大学，2004.

[3] 刘刚. 基于 BP 神经网络的隧道围岩稳定性分类的研究与工程应用[D]. 合肥：合肥工业大学，2007.

[4] 徐祥合. 基于 BP 神经网络的客户分类方法研究[D]. 南京：南京航空航天大学，2004.

[5] 刘旭生. 基于人工神经网络的森林植被遥感分类研究[D]. 北京：北京林业大学，2004.

[6] 韩立群. 人工神经网络理论、设计及应用[M]. 北京：化学工业出版社，2002.

[7] 余立雪. 神经网络与实例学习[M]. 北京：中国铁道出版社，1996.

[8] 周志华，曹存根. 神经网络及其应用[M]. 北京：清华大学出版社，2004.

第 2 章　BP 神经网络的非线性系统建模

——非线性函数拟合

2.1　案例背景

在工程应用中经常会遇到一些复杂的非线性系统，这些系统状态方程复杂，难以用数学方法准确建模。在这种情况下，可以建立 BP 神经网络表达这些非线性系统。该方法把未知系统看成是一个黑箱，首先用系统输入输出数据训练 BP 神经网络，使网络能够表达该未知函数，然后就可以用训练好的 BP 神经网络预测系统输出。

本章拟合的非线性函数为

$$y = x_1^2 + x_2^2 \tag{2-1}$$

该函数的图形如图 2-1 所示。

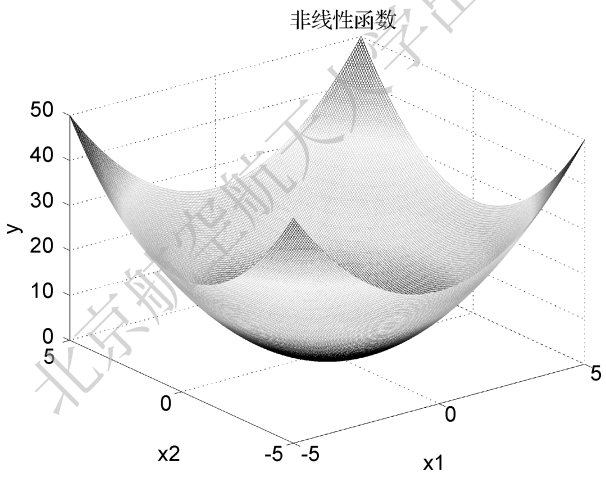

图 2-1　非线性函数图形

2.2　模型建立

基于 BP 神经网络的非线性函数拟合算法流程可以分为 BP 神经网络构建、BP 神经网络训练和 BP 神经网络预测三步，如图 2-2 所示。

BP 神经网络构建根据拟合非线性函数特点确定 BP 神经网络结构，由于该非线性函数有两个输入参数，一个输出参数，所以 BP 神经网络结构为 2—5—1，即输入层有 2 个节点，隐含层有 5 个节点，输出层有 1 个节点。

BP 神经网络训练用非线性函数输入输出数据训练神经网络，使训练后的网络能够预测非线性函数输出。从非线性函数中随机得到 2 000 组输入输出数据，从中随机选择 1 900 组作为

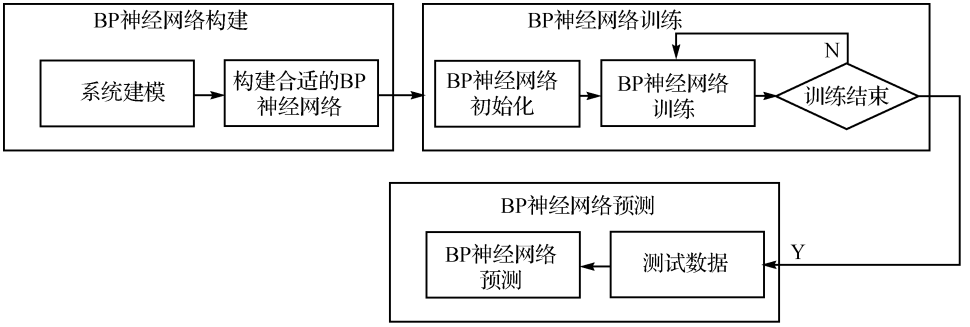

图 2-2　算法流程

训练数据，用于网络训练，100 组作为测试数据，用于测试网络的拟合性能。

神经网络预测用训练好的网络预测函数输出，并对预测结果进行分析。

2.3　MATLAB 实现

根据 BP 神经网络理论，用 MATLAB 软件编程实现基于 BP 神经网络的非线性拟合算法。

2.3.1　BP 神经网络工具箱函数

MATLAB 软件中包含 MATLAB 神经网络工具箱。它是以人工神经网络理论为基础，用 MATLAB 语言构造出了该理论所涉及的公式运算、矩阵操作和方程求解等大部分子程序以用于神经网络的设计和训练。用户只需根据自己的需要调用相关的子程序，即可以完成包括网络结构设计、权值初始化、网络训练及结果输出等在内的一系列工作，免除编写复杂庞大程序的困扰。目前，MATLAB 神经网络工具箱包括的网络有感知器、线性网络、BP 神经网络、径向基网络、自组织网络和回归网络等。BP 神经网络主要用到 newff，sim 和 train 3 个神经网络函数，各函数解释如下。

1. newff：BP 神经网络参数设置函数

函数功能：构建一个 BP 神经网络。

函数形式：net = newff(P,T,S,TF,BTF,BLF,PF,IPF,OPF,DDF)

P：输入数据矩阵。

T：输出数据矩阵。

S：隐含层节点数。

TF：节点传递函数，包括硬限幅传递函数 hardlim，对称硬限幅传递函数 hardlims，线性传递函数 purelin，正切 S 型传递函数 tansig，对数 S 型传递函数 logsig。

BTF：训练函数，包括梯度下降 BP 算法训练函数 traingd，动量反传的梯度下降 BP 算法训练函数 traingdm，动态自适应学习率的梯度下降 BP 算法训练函数 traingda，动量反传和动态自适应学习率的梯度下降 BP 算法训练函数 traingdx，Levenberg_Marquardt 的 BP 算法训练函数 trainlm。

BLF：网络学习函数，包括 BP 学习规则 learngd，带动量项的 BP 学习规则 learngdm。

PF:性能分析函数,包括均值绝对误差性能分析函数 mae,均方差性能分析函数 mse。

IPF:输入处理函数。

OPF:输出处理函数。

DDF:验证数据划分函数。

一般在使用过程中设置前面 6 个参数,后面 4 个参数采用系统默认参数。

2. train:BP 神经网络训练函数

函数功能:用训练数据训练 BP 神经网络。

函数形式:[net,tr] = train(NET,X,T,Pi,Ai)

NET:待训练网络。

X:输入数据矩阵。

T:输出数据矩阵。

Pi:初始化输入层条件。

Ai:初始化输出层条件。

net:训练好的网络。

tr:训练过程记录。

一般在使用过程中设置前面 3 个参数,后面 2 个参数采用系统默认参数。

3. sim:BP 神经网络预测函数

函数功能:用训练好的 BP 神经网络预测函数输出。

函数形式:y=sim(net,x)

net:训练好的网络。

x:输入数据。

y:网络预测数据。

2.3.2 数据选择和归一化

根据非线性函数方程随机得到该函数的 2 000 组输入输出数据,将数据存储在 data.mat 文件中,input 是函数输入数据,output 是函数输出数据。从输入输出数据中随机选取 1 900 组数据作为网络训练数据,100 组数据作为网络测试数据,并对训练数据进行归一化处理。

```
%清空环境变量
clc
clear

%下载输入输出数据
load data input output

%随机选择 1900 组训练数据和 100 组预测数据
k = rand(1,2000);
[m,n] = sort(k);
input_train = input(n(1:1900),:)';
output_train = output(n(1:1900),:)';
input_test = input(n(1901:2000),:)';
```

```
output_test = output(n(1901:2000),:)';

%训练数据归一化
[inputn,inputps] = mapminmax(input_train);
[outputn,outputps] = mapminmax(output_train);
```

2.3.3 BP 神经网络训练

用训练数据训练 BP 神经网络,使网络对非线性函数输出具有预测能力。

```
%BP 神经网络构建
net = newff(inputn,outputn,5);

%网络参数配置(迭代次数,学习率,目标)
net.trainParam.epochs = 100;
net.trainParam.lr = 0.1;
net.trainParam.goal = 0.00004;

%BP 神经网络训练
net = train(net,inputn,outputn);
```

2.3.4 BP 神经网络预测

用训练好的 BP 神经网络预测非线性函数输出,并通过 BP 神经网络预测输出和期望输出分析 BP 神经网络的拟合能力。

```
%预测数据归一化
inputn_test = mapminmax('apply',input_test,inputps);

%BP 神经网络预测输出
an = sim(net,inputn_test);

%输出结果反归一化
BPoutput = mapminmax('reverse',an,outputps);

%网络预测结果图形
figure(1)
plot(BPoutput,':og')
hold on
plot(output_test,'-*');
legend('预测输出','期望输出')
title('BP 网络预测输出','fontsize',12)
ylabel('函数输出','fontsize',12)
xlabel('样本','fontsize',12)

%网络预测误差图形
```

```
figure(2)
plot(error,'- *')
title('BP 网络预测误差','fontsize',12)
ylabel('误差','fontsize',12)
xlabel('样本','fontsize',12)
```

2.3.5 结果分析

用训练好的 BP 神经网络预测函数输出，预测结果如图 2-3 所示。

BP 神经网络预测输出和期望输出的误差如图 2-4 所示。

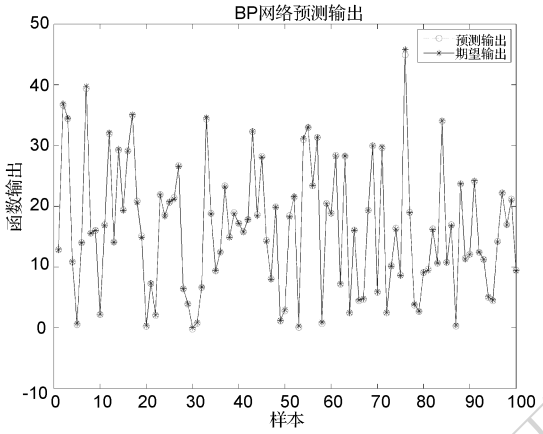

图 2-3 BP 神经网络预测

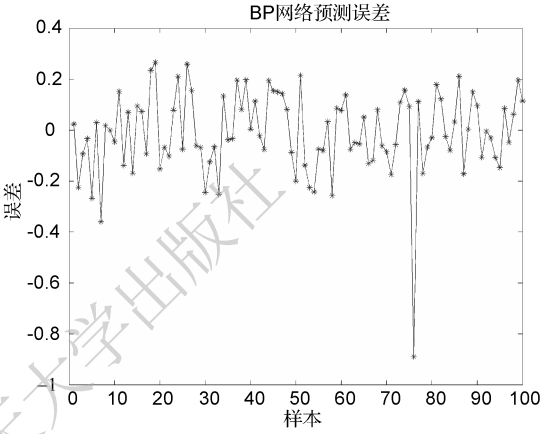

图 2-4 BP 神经网络预测误差

从图 2-3 和图 2-4 可以看出，虽然 BP 神经网络具有较高的拟合能力，但是网络预测结果仍有一定误差，某些样本点的预测误差较大。后面案例中将讨论 BP 神经网络优化算法，以得到更好的预测结果。

2.4 案例扩展

2.4.1 多隐含层 BP 神经网络

BP 神经网络由输入层、隐含层和输出层组成，隐含层根据层数又可以分为单隐含层和多隐含层。多隐含层由多个单隐含层组成，同单隐含层相比，多隐含层泛化能力强，预测精度高，但是训练时间较长。隐含层层数的选择要从网络精度和训练时间上综合考虑，对于较简单的映射关系，在网络精度达到要求的情况下，可以选择单隐含层，以求加快速度；对于复杂的映射关系，则可以选择多隐含层，以期提高网络的预测精度。

MATLAB 神经网络工具箱中的 newff 函数可以方便地构建包含多个隐含层的 BP 神经网络，其调用函数如下：

```
net = newff(P,T,S,TF,BTF,BLF,PF,IPF,OPF,DDF)
```

根据 newff 的帮助文件可知 newff 函数的第三个参数 S 的解释如下。

Si—Sizes of N－1 hidden layers, S1 to S(N－1), default = [].　(Si 为第 i 个隐含层节点数,i=1:N－1)

从英文帮助可知,S 是不同隐含层包含的节点数向量,通过配置 S 向量,可以方便地得到包含多个隐含层的 BP 神经网络,如下面语句:

```
net = newff(inputn,outputn,[5,5]);
```

该语句构建了双隐含层 BP 神经网络,每个隐含层的节点数都是 5,程序运行时显示的网络结构和运行过程如图 2－5 所示。

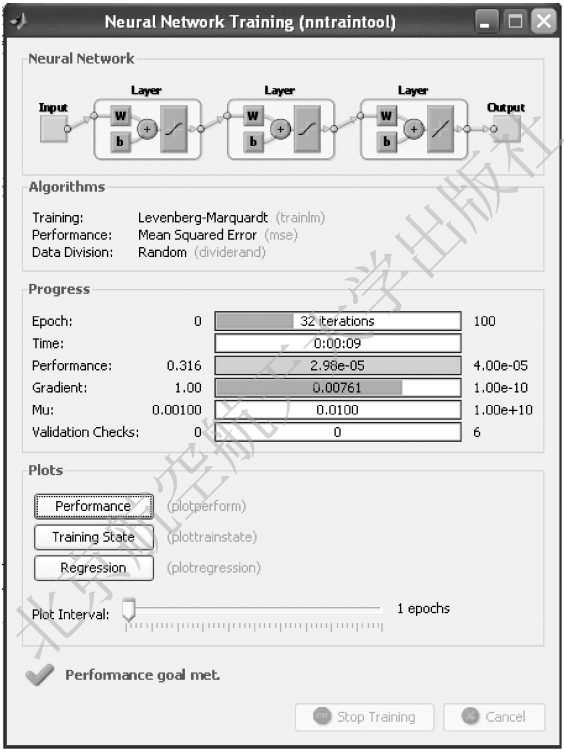

图 2－5　双隐含层 BP 神经网络

从运行时间和预测精度两个方面比较单隐含层 BP 神经网络和双隐含层 BP 神经网络的性能,网络结构相同,训练迭代都为 100 次,比较 10 次预测结果平均值,比较结果如表 2－1 所列。

表 2－1　BP 神经网络预测结果比较

网络类别	预测误差百分比	均方误差	运行时间/s
单隐含层 BP 神经网络	1.64%	0.007 6	8.005 3
双隐含层 BP 神经网络	1.60%	0.001 6	9.859 2

从表 2－1 可以看出，双隐含层 BP 神经网络同单隐含层 BP 神经网络相比，预测精度有所提高，但是运行时间有所增加。

2.4.2 隐含层节点数

BP 神经网络构建时应注意隐含层节点数的选择，如果隐含层含节点数太少，BP 神经网络不能建立复杂的映射关系，网络预测误差较大。但是如果节点数过多，网络学习时间增加，并且可能出现“过拟合”现象，就是训练样本预测准确，但是其他样本预测误差较大。不同隐含层节点数 BP 神经网络预测误差如表 2－2 所列。

表 2－2 不同隐含层节点数 BP 神经网络预测误差

隐含层节点数	3	4	5	6	7	8	9
相对误差百分比	5.46%	1.75%	1.64%	0.32%	0.31%	0.29%	0.08%
均方误差	0.009 4	0.013 1	0.007 6	0.001 2	0.000 4	0.000 2	0.000 1

由于本案例拟合的非线性函数较为简单，所以 BP 神经网络预测误差随着节点数的增加而不断减少，但是对于复杂问题来说，网络预测误差随节点数增加一般呈现先减少后增加的趋势。

2.4.3 训练数据对预测精度影响

神经网络预测的准确性和训练数据的多少有较大的关系，尤其对于一个多输入多输出的网络，如果缺乏足够多的网络训练数据，网络预测值可能存在较大的误差。

笔者曾经做过一个 BP 神经网络预测实例，该实例通过建立一个 4 输入、5 输出 BP 神经网络预测实验结果。网络训练数据来自于真实实验，由于实验过程复杂，故只取到 84 组数据，选择其中 80 组数据作为 BP 神经网络训练数据，其余 4 组数据作为测试数据，训练后的 BP 神经网络预测结果如表 2－3 所列。

表 2－3 BP 神经网络预测结果

预测值					期望值				
输出 1	输出 2	输出 3	输出 4	输出 5	输出 1	输出 2	输出 3	输出 4	输出 5
0.515 7	2.850 1	1.796 8	1.513 3	2.038 2	2.800 0	2.000 0	1.200 0	2.800 0	2.000 0
1.467 1	1.821 5	1.319 3	0.703 6	1.482 5	2.200 0	1.800 0	1.400 0	0.6000	0.000 0
1.348 8	1.715 8	1.741 1	2.073 6	1.259 2	1.600 0	1.600 0	1.600 0	1.6000	1.600 0
1.216 0	1.773 2	1.706 6	1.595 9	0.510 1	1.000 0	1.400 0	1.800 0	2.600 0	3.000 0

从表 2－3 可以看出，由于缺乏训练数据，BP 神经网络没有得到充分训练，BP 神经网络预测值和期望值之间误差较大。

笔者曾经做过一个类似的预测问题，该问题的目的是构建一个 4 输入 4 输出的 BP 神经网络预测系统输出，训练数据来自于模型仿真结果。由于该模型可以通过软件模拟，所以得到多组数据，选择 1 500 组数据训练网络，最后网络预测值同期望值比较接近。

2.4.4 节点转移函数

MATLAB 神经网络工具箱中 newff 函数提供了几种节点转移函数，主要包括以下三种。

1）logsig 函数：

$$y = 1/[1 - \exp(-x)] \tag{2-2}$$

2）tansig 函数：

$$y = 2/[1 - \exp(-2x)] - 1 \tag{2-3}$$

3）purelin 函数：

$$y = x \tag{2-4}$$

在网络结构和权值、阈值相同的情况下，BP 神经网络预测误差和均方误差、输出层节点转移函数的关系如表 2-4 所列。

表 2-4 不同转移函数对应预测误差

隐含层函数	输出层函数	误差百分比	均方误差
logsig	tansig	40.63%	0.902 5
logsig	purelin	0.08%	0.000 1
logsig	logsig	352.65%	181.251 1
tansig	tansig	31.90%	1.173 3
tansig	logsig	340.90%	162.969 8
tansig	purelin	1.70%	0.010 7
purelin	logsig	343.36%	143.763 34
purelin	tansig	120.08%	113.028 1
purelin	purelin	196.49%	99.012 1

从表 2-4 可以看出，隐含层和输出层函数的选择对 BP 神经网络预测精度有较大影响。一般隐含层节点转移函数选用 logsig 函数或 tansig 函数，输出层节点转移函数选用 tansig 或 purelin 函数。

2.4.5 网络拟合的局限性

BP 神经网络虽然具有较好的拟合能力，但其拟合能力不是绝对的，对于一些复杂系统，BP 神经网络预测结果会存在较大误差。比如对于

$$y = (x_1^2 + x_2^2)^{0.25}\{\sin^2[50(x_1^2 + x_2^2)^{0.1}] + 1\} \tag{2-5}$$

其函数图形如图 2-6 所示。

随机选取该函数 2 000 组输入输出数据，从中取 1 900 组数据训练网络，100 组数据测试网络拟合能力。采用单隐含层 BP 神经网络，网络结构为 2—5—1，网络训练 100 次后预测函数输出，预测结果如图 2-7 所示。

从图 2-7 可以看出，对于复杂的非线性系统，BP 神经网络预测误差较大。该例说明 BP 神经网络的拟合能力具有局限性。

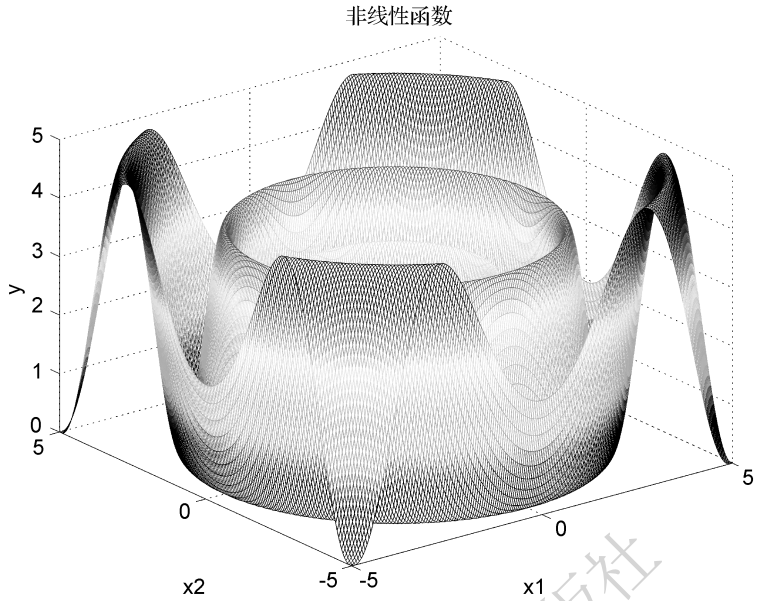

图 2-6　复杂函数图形

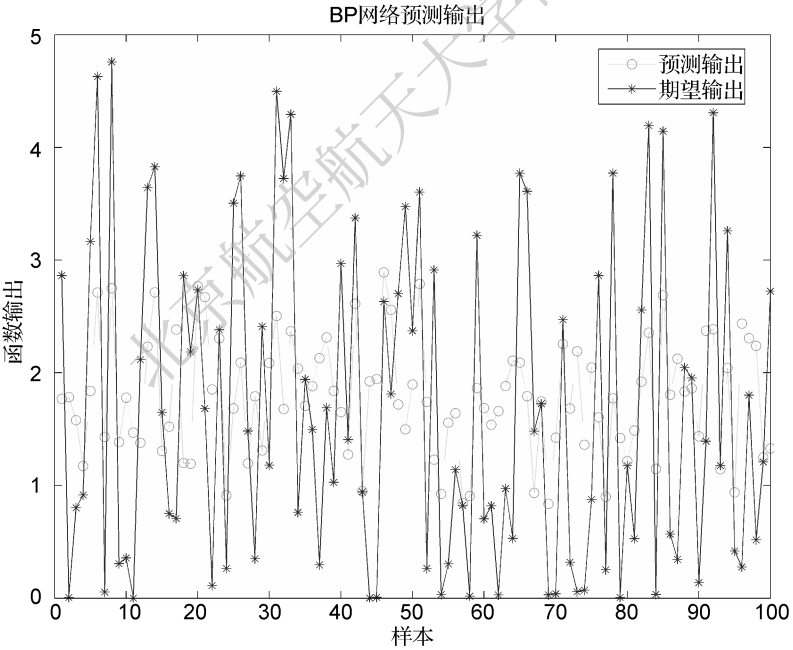

图 2-7　BP 神经网络预测结果

参考文献

[1] 邓伟. BP 神经网络构建与优化的研究及其在医学统计中的应用[D]. 上海:复旦大学,2002.

[2] 林盾,陈俐. BP 神经网络在模拟非线性系统输出中的应用[J]. 武汉理工大学学报,2003,27

(5):731－734.

[3] 吕蝉.基于BP神经网络的短期负荷预测[D].武汉:华中科技大学,2007.

[4] 陈敏.基于BP神经网络的混沌时间序列预测模型研究[D].长沙:中南大学,2007.

[5] 林琳.基于BP神经网络的网格性能预测[D].长春:吉林大学,2004.

[6] 雷晓云,张丽霞,梁新平.基于MATLAB工具箱的BP神经网路年径流量预测模型研究[J].水文,2008,28(1):43－48.

[7] 刘双.基于Matlab神经网络工具箱的电力负荷组合预测模型[J].电力自动化设备,2003,23(3),59－61.

第 3 章　遗传算法优化 BP 神经网络

——非线性函数拟合

3.1　案例背景

3.1.1　遗传算法原理

遗传算法(Genetic Algorithms)是 1962 年由美国 Michigan 大学 Holland 教授提出的模拟自然界遗传机制和生物进化论而成的一种并行随机搜索最优化方法。它把自然界“优胜劣汰,适者生存”的生物进化原理引入优化参数形成的编码串联群体中,按照所选择的适应度函数并通过遗传中的选择、交叉和变异对个体进行筛选,使适应度值好的个体被保留,适应度差的个体被淘汰,新的群体既继承了上一代的信息,又优于上一代。这样反复循环,直至满足条件。遗传算法基本的操作分为:

1. 选择操作

选择操作是指从旧群体中以一定概率选择个体到新群体中,个体被选中的概率跟适应度值有关,个体适应度值越好,被选中的概率越大。

2. 交叉操作

交叉操作是指从个体中选择两个个体,通过两个染色体的交换组合,来产生新的优秀个体。交叉过程为从群体中任选两个染色体,随机选择一点或多点染色体位置进行交换。交叉操作如图 3-1 所示。

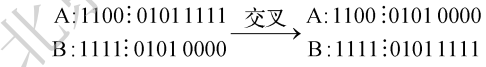

图 3-1　交叉操作

3. 变异操作

变异操作是指从群体中任选一个个体,选择染色体中的一点进行变异以产生更优秀的个体。变异操作如图 3-2 所示。

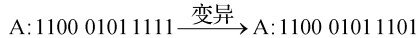

图 3-2　变异操作

遗传算法具有高效启发式搜索、并行计算等特点,目前已经应用在函数优化、组合优化以及生产调度等方面。

3.1.2　遗传算法的基本要素

遗传算法的基本要素包括染色体编码方法、适应度函数、遗传操作和运行参数。

其中染色体编码方法是指个体的编码方法，目前包括二进制法、实数法等。二进制法是指把个体编码成为一个二进制串，实数法是指把个体编码成为一个实数串。

适应度函数是指根据进化目标编写的计算个体适应度值的函数，通过适应度函数计算每个个体的适应度值，提供给选择算子进行选择。

遗传操作是指选择操作、交叉操作和变异操作。

运行参数是遗传算法在初始化时确定的参数，主要包括群体大小 M，遗传代数 G，交叉概率 P_c 和变异概率 P_m。

3.1.3 拟合函数

本案例拟合的非线性函数为

$$y = x_1^2 + x_2^2 \tag{3-1}$$

该函数的图形如图 3-3 所示。

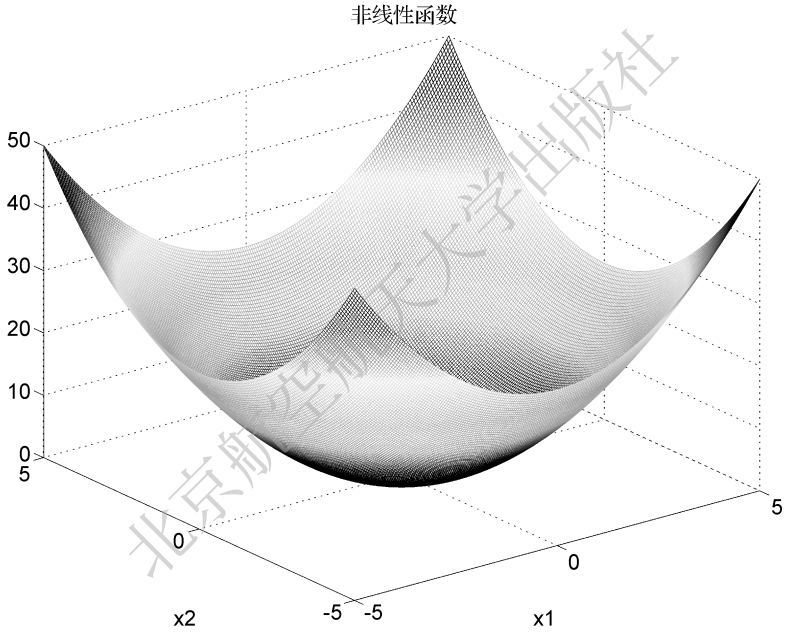

图 3-3 非线性函数图形

3.2 模型建立

3.2.1 算法流程

遗传算法优化 BP 神经网络算法流程如图 3-4 所示。

遗传算法优化 BP 神经网络分为 BP 神经网络结构确定、遗传算法优化和 BP 神经网络预测 3 个部分。其中，BP 神经网络结构确定部分根据拟合函数输入输出参数个数确定 BP 神经网络结构，进而确定遗传算法个体的长度。遗传算法优化使用遗传算法优化 BP 神经网络的权值和阈值，种群中的每个个体都包含了一个网络所有权值和阈值，个体通过适应度函数计算

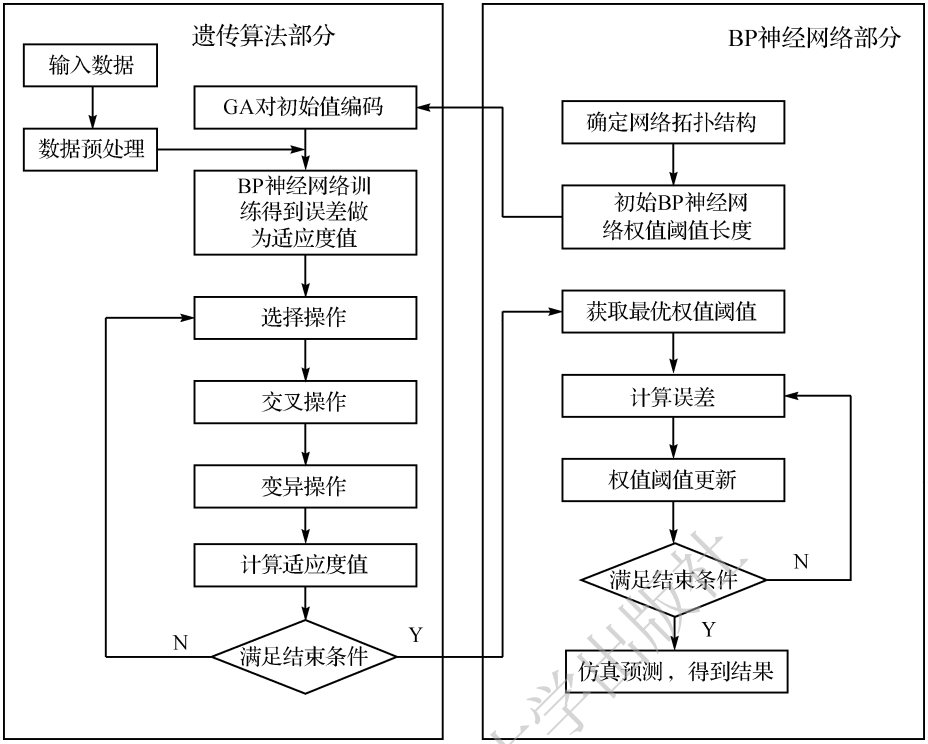

图 3-4　算法流程

个体适应度值，遗传算法通过选择、交叉和变异操作找到最优适应度值对应个体。BP 神经网络预测用遗传算法得到最优个体对网络初始权值和阈值赋值，网络经训练后预测函数输出。

本案例中，由于拟合非线性函数有 2 个输入参数、1 个输出参数，所以设置的 BP 神经网络结构为 2—5—1，即输入层有 2 个节点，隐含层有 5 个节点，输出层有 1 个节点，共有 2×5＋5×1＝15 个权值，5＋1＝6 个阈值，所以遗传算法个体编码长度为 16＋5＝21。从非线性函数中随机得到 2 000 组输入输出数据，从中随机选择 1 900 组作为训练数据，用于网络训练，100 组作为测试数据。把训练数据预测误差绝对值和作为个体适应度值，个体适应度值越小，该个体越优。

3.2.2　遗传算法实现

遗传算法优化 BP 神经网络是用遗传算法来优化 BP 神经网络的初始权值和阈值，使优化后的 BP 神经网络能够更好地预测函数输出。遗传算法优化 BP 神经网络的要素包括种群初始化、适应度函数、选择操作、交叉操作和变异操作。

1. 种群初始化

个体编码方法为实数编码，每个个体均为一个实数串，由输入层与隐含层连接权值、隐含层阈值、隐含层与输出层连接权值以及输出层阈值 4 部分组成。个体包含了神经网络全部权值和阈值，在网络结构已知的情况下，就可以构成一个结构、权值、阈值确定的神经网络。

2. 适应度函数

根据个体得到 BP 神经网络的初始权值和阈值，用训练数据训练 BP 神经网络后预测系统

输出，把预测输出和期望输出之间的误差绝对值和 E 作为个体适应度值 F，计算公式为

$$F = k\left(\sum_{i=1}^{n} abs(y_i - o_i)\right) \tag{3-2}$$

式中，n 为网络输出节点数；y_i 为 BP 神经网络第 i 个节点的期望输出；o_i 为第 i 个节点的预测输出；k 为系数。

3. 选择操作

遗传算法选择操作有轮盘赌法、锦标赛法等多种方法，本案例选择轮盘赌法，即基于适应度比例的选择策略，每个个体 i 的选择概率 p_i 为

$$f_i = k/F_i \tag{3-3}$$

$$p_i = \frac{f_i}{\sum_{j=1}^{N} f_j} \tag{3-4}$$

式中，F_i 为个体 i 的适应度值，由于适应度值越小越好，所以在个体选择前对适应度值求倒数；k 为系数；N 为种群个体数目。

4. 交叉操作

由于个体采用实数编码，所以交叉操作方法采用实数交叉法，第 k 个染色体 a_k 和第 l 个染色体 a_l 在 j 位的交叉操作方法如下：

$$\left.\begin{aligned} a_{kj} &= a_{kj}(1-b) + a_{lj}b \\ a_{lj} &= a_{lj}(1-b) + a_{kj}b \end{aligned}\right\} \tag{3-5}$$

式中，b 是[0,1]间的随机数。

5. 变异操作

选取第 i 个个体的第 j 个基因 a_{ij} 进行变异，变异操作方法如下：

$$a_{ij} = \begin{cases} a_{ij} + (a_{ij} - a_{\max}) * f(g) & r \geqslant 0.5 \\ a_{ij} + (a_{\min} - a_{ij}) * f(g) & r < 0.5 \end{cases} \tag{3-6}$$

式中，$a_{\max}$ 为基因 a_{ij} 的上界；$a_{\min}$ 为基因 a_{ij} 的下界；$f(g) = r_2(1 - g/G_{\max})$；$r_2$ 为一个随机数；g 为当前迭代次数；$G_{\max}$ 是最大进化次数；r 为[0,1]间的随机数。

3.3 编程实现

根据遗传算法和 BP 神经网络理论，在 MATLAB 软件中编程实现基于遗传算法优化的 BP 神经网络非线性系统拟合算法。遗传算法参数设置为：种群规模为 10，进化次数为 50 次，交叉概率为 0.4，变异概率为 0.2。MATLAB 代码如下。

3.3.1 适应度函数

适应度函数用训练数据训练 BP 神经网络，并且把训练数据预测误差作为个体适应度值。

```
function error = fun(x,inputnum,hiddennum,outputnum,net,inputn,outputn)
%该函数用来计算适应度值
%x           input      个体
%inputnum    input      输入层节点数
%outputnum   input      隐含层节点数
```

```
    % net          input      网络
    % inputn       input      训练输入数据
    % outputn      input      训练输出数据

    % error        output     个体适应度值

    % BP 神经网络初始权值和阈值,x 为个体
    w1 = x(1:inputnum * hiddennum);
    B1 = x(inputnum * hiddennum + 1:inputnum * hiddennum + hiddennum);
    w2 = x(inputnum * hiddennum + hiddennum + 1:inputnum * hiddennum + hiddennum + hiddennum * output-
num);
    B2 = x(inputnum * hiddennum + hiddennum + hiddennum * outputnum + 1:inputnum * hiddennum + hidden-
num + hiddennum * outputnum + outputnum);

    net.iw{1,1} = reshape(w1,hiddennum,inputnum);
    net.lw{2,1} = reshape(w2,outputnum,hiddennum);
    net.b{1} = reshape(B1,hiddennum,1);
    net.b{2} = B2;

    % BP 神经网络构建
    net = newff(inputn,outputn,hiddennum);
    net.trainParam.epochs = 20;
    net.trainParam.lr = 0.1;
    net.trainParam.goal = 0.00001;
    net.trainParam.show = 100;
    net.trainParam.showWindow = 0;


    % BP 神经网络训练
    net = train(net,inputn,outputn);

    % BP 神经网络预测
    an = sim(net,inputn);

    % 预测误差和作为个体适应度值
    error = sum(abs(an - outputn));
```


3.3.2 选择操作

选择操作采用轮盘赌法从种群中选择适应度好的个体组成新种群。

```
    function ret = select(individuals,sizepop)
    % 该函数用于进行选择操作
    % individuals input   种群信息
    % sizepop     input     种群规模
    % ret         output    选择后的新种群
```

```
%求适应度值倒数
fitness1 = 10./individuals.fitness;  %individuals.fitness 为个体适应度值

%个体选择概率
sumfitness = sum(fitness1);
sumf = fitness1./sumfitness;

%采用轮盘赌法选择新个体
index = [];
for i = 1:sizepop     %sizepop 为种群数
    pick = rand;
    while pick == 0
        pick = rand;
    end
    for i = 1:sizepop
        pick = pick - sumf(i);
        if pick<0
            index = [index i];
            break;
        end
    end
end

%新种群
individuals.chrom = individuals.chrom(index,:);    %individuals.chrom 为种群中个体
individuals.fitness = individuals.fitness(index);
ret = individuals;
```

3.3.3 交叉操作

交叉操作从种群中选择两个个体，按一定概率交叉得到新个体。

```
function ret = Cross(pcross,lenchrom,chrom,sizepop,bound)
%该函数用于进行交叉操作
% pcorss                input     交叉概率
% lenchrom              input     个体长度
% chrom                 input     种群个体
% sizepop               input     种群规模
% ret                   output    交叉后的新种群

for i = 1:sizepop %sizepop 为种群个体数目

    %选择选择两个个体
    pick = rand(1,2);
    while prod(pick) == 0
```

```
        pick = rand(1,2);
    end
    index = ceil(pick. * sizepop);

%判断是否交叉
    pick = rand;
    while pick == 0
        pick = rand;
    end
    if pick>pcross      %pcross 为交叉概率
        continue;
    end
    flag = 0;
    while flag == 0
        %选择交叉位置
        pick = rand;
        while pick == 0
            pick = rand;
        end
        pos = ceil(pick. * sum(lenchrom));   %lenchrom  为个体长度

        %个体交叉
        pick = rand;
        v1 = chrom(index(1),pos);
        v2 = chrom(index(2),pos);
        chrom(index(1),pos) = pick * v2 + (1 - pick) * v1;
        chrom(index(2),pos) = pick * v1 + (1 - pick) * v2;

        %测试新个体是否满足约束要求
        flag1 = test(lenchrom,bound,chrom(index(1),:));
        flag2 = test(lenchrom,bound,chrom(index(2),:));
        if   flag1 * flag2 == 0
            flag = 0;
        else flag = 1;
        end
    end
 end
ret = chrom;
```

3.3.4 变异操作

变异操作从种群中随机选择一个个体,按一定概率变异得到新个体。

```
function ret = Mutation(pmutation,lenchrom,chrom,sizepop,num,maxgen,bound)
%该函数用于完成变异操作
% pcorss                    input      变异概率
```

```
% lenchrom                 input     个体长度
% chrom                    input     种群个体
% sizepop                  input     种群规模
% bound                    input     个体上界和下界
% maxgen                   input     最大迭代次数
% num                      input     当前迭代次数
% ret                      output    交叉后地新种群

for i = 1:sizepop     % sizepop 为种群数

    % 变异概率
    pick = rand;
    while pick == 0
        pick = rand;
    end
    index = ceil(pick * sizepop);

    % 判断是否变异
    pick = rand;
    if pick>pmutation    % pmutation 为变异概率
        continue;
    end
    flag = 0;
    while flag == 0
        % 随机选择变异位置
        pick = rand;
        while pick == 0
            pick = rand;
        end
        pos = ceil(pick * sum(lenchrom));   % lengchrom 为个体长度
        % 变异操作
        v = chrom(i,pos);
        v1 = v - bound(pos,1);
        v2 = bound(pos,2) - v;
        pick = rand;
        fg = (rand * (1 - num/maxgen))^2;    % num 遗传算法当前迭代次数,maxgen 总迭代次数
        if pick>0.5
            chrom(i,pos) = chrom(i,pos) + (bound(pos,2) - chrom(i,pos)) * fg;
        else
            chrom(i,pos) = chrom(i,pos) - (chrom(i,pos) - bound(pos,1)) * fg;
        end
        flag = test(lenchrom,bound,chrom(i,:));       % 新个体是否满足约束要求
    end
end
ret = chrom;
```

3.3.5 遗传算法主函数

遗传算法主函数流程为

步骤 1:随机初始化种群;

步骤 2:计算种群适应度值,从中找出最优个体;

步骤 3:选择操作;

步骤 4:交叉操作;

步骤 5:变异操作;

步骤 6:判断进化是否结束,若否,则返回步骤 2。

主函数 MATLAB 代码主要部分如下。其中非线性函数的输入输出数据都在 data.mat 文件中,input 矩阵为输入数据,output 矩阵为输出数据。

```
%清空环境变量
clc
clear

%读取数据
load data input output

%网络结构
inputnum = 2;
hiddennum = 5;
outputnum = 1;

%取训练数据和预测数据
input_train = input(1:1900,:)';
input_test = input(1901:2000,:)';
output_train = output(1:1900)';
output_test = output(1901:2000)';

%数据归一化
[inputn,inputps] = mapminmax(input_train);
[outputn,outputps] = mapminmax(output_train);

%构建网络
net = newff(inputn,outputn,hiddennum);

%遗传算法参数初始化
maxgen = 50;                          %迭代次数
sizepop = 10;                         %种群规模
pcross = [0.4];                       %交叉概率
pmutation = [0.2];                    %变异概率

%节点总数
```

```
numsum = inputnum * hiddennum + hiddennum + hiddennum * outputnum + outputnum;

lenchrom = ones(1,numsum);                              % 个体长度
bound = [ - 3 * ones(numsum,1) 3 * ones(numsum,1)];     % 个体范围

% 种群信息定义为结构体
individuals = struct('fitness',zeros(1,sizepop),'chrom',[]);
avgfitness = [];                        % 每代平均适应度值
bestfitness = [];                       % 每代最佳适应度值
bestchrom = [];                         % 最优个体

% 计算个体适应度值
for i = 1:sizepop
    % 个体初始化
    individuals.chrom(i,:) = Code(lenchrom,bound);

    % 计算个体适应度值
    x = individuals.chrom(i,:);
    individuals.fitness(i) = fun(x,inputnum,hiddennum,outputnum,net,inputn,outputn);
end

% 迭代寻优
for i = 1:maxgen

    % 选择操作
    individuals = Select(individuals,sizepop);
    % 交叉操作
    individuals.chrom = Cross(pcross,lenchrom,individuals.chrom,sizepop,bound);
    % 变异操作
    individuals.chrom = Mutation(pmutation,lenchrom,individuals.chrom,sizepop,i,
    maxgen,bound);

    % 计算适应度值
    for j = 1:sizepop
        x = individuals.chrom(j,:); % 个体
        individuals.fitness(j) = fun(x,inputnum,hiddennum,outputnum,net,inputn,output n);
        end

    % 寻找最优最差个体
    [newbestfitness,newbestindex] = min(individuals.fitness);
    [worestfitness,worestindex] = max(individuals.fitness);

    % 最优个体更新
    if bestfitness>newbestfitness
        bestfitness = newbestfitness;
```

```
            bestchrom = individuals.chrom(newbestindex,:);
        end
        individuals.chrom(worestindex,:) = bestchrom;
        individuals.fitness(worestindex) = bestfitness;

        %记录最优个体适应度值和平均适应度值
        avgfitness = sum(individuals.fitness)/sizepop;
        trace = [trace;avgfitness bestfitness];

    end
```

3.3.6 遗传算法优化的 BP 神经网络函数拟合

把遗传算法得到的最优个体赋给 BP 神经网络,用该网络拟合非线性函数。

```
    %把最优个体 x 赋给 BP 神经网络权值和阈值
    x = bestchrom
    w1 = x(1:inputnum * hiddennum);
    B1 = x(inputnum * hiddennum + 1:inputnum * hiddennum + hiddennum);
    w2 = x(inputnum * hiddennum + hiddennum + 1:inputnum * hiddennum + hiddennum + hiddennum * output-
num);
    B2 = x(inputnum * hiddennum + hiddennum + hiddennum * outputnum + 1:inputnum * hiddennum + hidden-
num + hiddennum * outputnum + outputnum);

    net.iw{1,1} = reshape(w1,hiddennum,inputnum);
    net.lw{2,1} = reshape(w2,outputnum,hiddennum);
    net.b{1} = reshape(B1,hiddennum,1);
    net.b{2} = B2;
    %BP 神经网络参数
    net.trainParam.epochs = 100;
    net.trainParam.lr = 0.1;
    %net.trainParam.goal = 0.00001;

    %BP 神经网络训练
    [net,per2] = train(net,inputn,outputn);

    %BP 神经网络预测
    inputn_test = mapminmax('apply',input_test,inputps);
    an = sim(net,inputn_test);

    %预测结果反归一化
    test_simu = mapminmax('reverse',an,outputps);
```

3.3.7 结果分析

遗传算法优化过程中最优个体适应度值变化如图 3-5 所示。

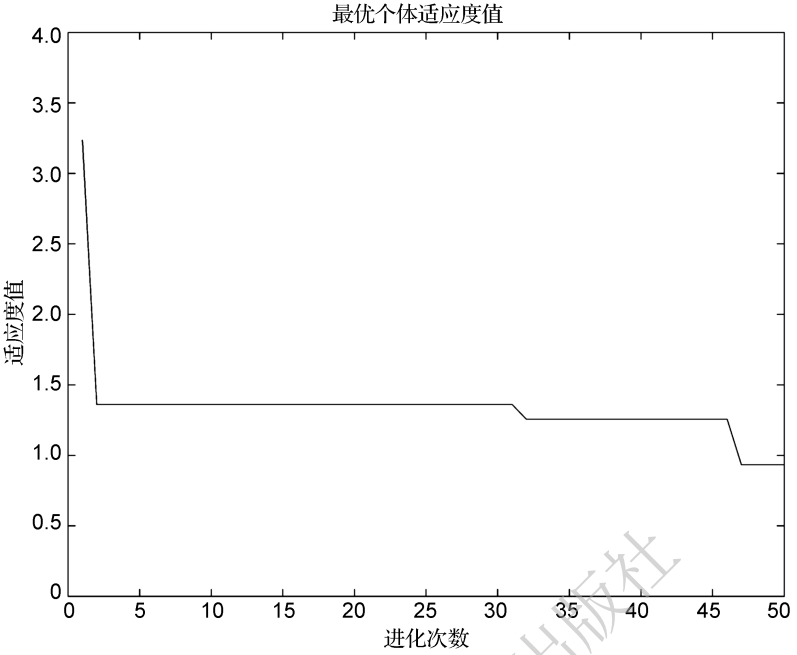

图 3-5　最优个体适应度值

遗传算法优化得到的 BP 神经网络最优初始权值和阈值如表 3.1 所列。

表 3.1　最优初始权值阈值

输入层隐含层间权值	0.331 6	−1.333 4	−2.548 8	0.534 6	1.922 5	0.528 1	0.202 2	−0.164 3	−1.057 1	−0.414 1
隐含层节点阈值	1.589 4	2.978 8	−1.409 9	1.752 4	−2.192 3	—	—	—	—	—
隐含层输出层间权值	−2.786 8	−0.276 3	0.917 9	−1.614 7	−2.131 4	—	—	—	—	—
输出层节点阈值	1.062 6	—	—	—	—	—	—	—	—	—

把最优初始权值和阈值赋给神经网络，用训练数据训练 100 次后预测非线性函数输出，预测误差如图 3-6 所示。

从图 3-6 可以看出，遗传算法优化的 BP 网络预测更加精确，并且遗传算法优化 BP 网络预测的均方误差为 $5.370\ 4\times10^{-5}$，而未优化 BP 网络的均方误差为 $1.887\ 6\times10^{-4}$，预测均方误差也得到了很大改善。

3.4　案例扩展

3.4.1　其他优化方法

遗传算法优化 BP 神经网络的目的是通过遗传算法得到更好的网络初始权值和阈值，其基本思想就是用个体代表网络的初始权值和阈值、个体值初始化的 BP 神经网络的预测误差作为该个体的适应度值，通过选择、交叉、变异操作寻找最优个体，即最优的 BP 神经网络初始权值。除了遗传算法之外，还可以采用粒子群算法、蚁群算法等优化 BP 神经网络初始权值。

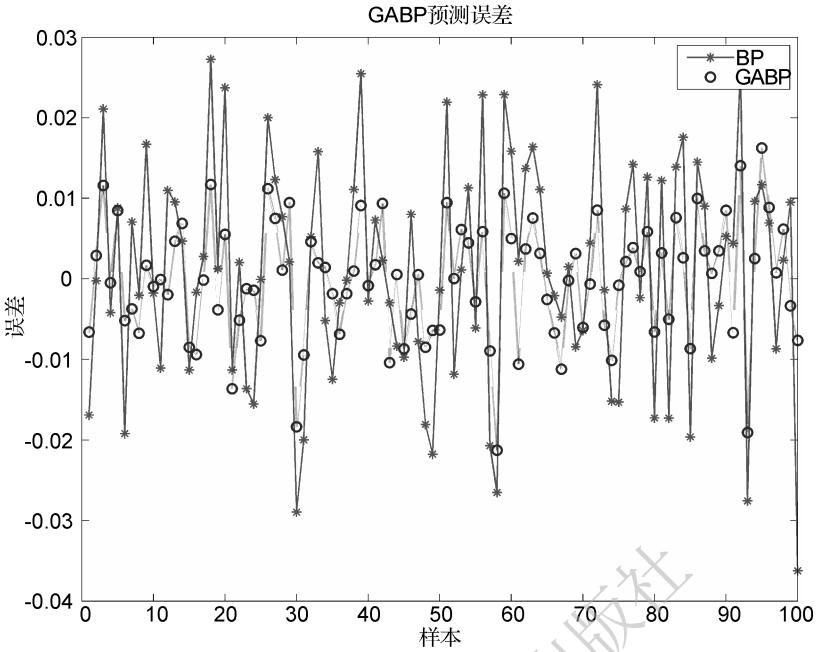

图 3-6　GA 优化 BP 网络预测误差

本案例同时实现了基于 PSO 算法(粒子群算法)的 BP 神经网络权值阈值优化,每个粒子代表了神经网络的权值和阈值,通过粒子寻优找到网络最佳的初始权值和阈值。粒子群算法具体操作方法见第 27 章,基本参数为:种群规模为 30,进化次数为 50,粒子群算法优化过程中最优个体适应度值变化过程如图 3-7 所示。

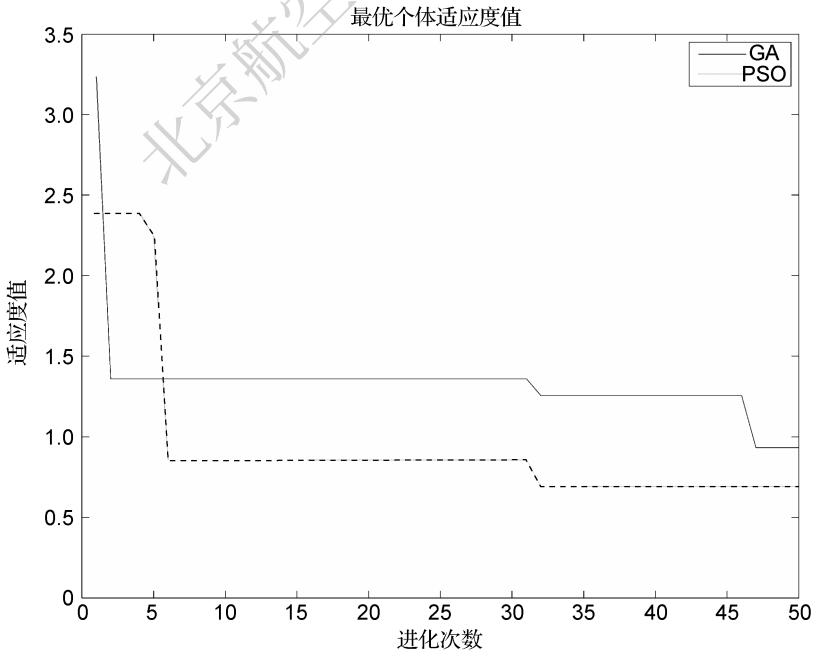

图 3-7　PSO 算法优化过程

把 PSO 算法得到的最优初始权值和阈值赋给神经网络，用训练数据训练 100 次后预测非线性函数输出，预测误差如图 3－8 所示。

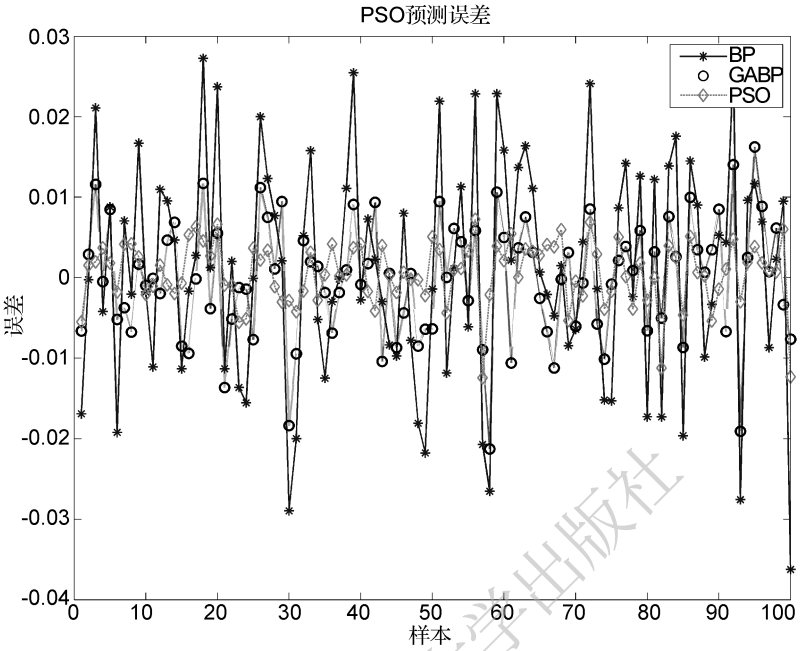

图 3－8　PSO 算法预测误差

从预测结果可以看出，基于 PSO 优化的 BP 神经网络预测误差更小，PSO 算法优化 BP 神经网络预测精度效果优于 GA 算法优化 BP 神经网络预测精度。

3.4.2　网络结构优化

有学者研究了基于遗传算法优化的 BP 神经网络结构优化算法，用遗传算法优化 BP 神经网络隐含层节点数目，对于时间序列预测问题，还可以用于优化输入层节点数。对于结构优化问题，种群个体采用二进制编码，适应度函数为预测误差，通过选择、交叉和变异操作得到 BP 神经网络最优结构。但是对于结构优化算法，由于权值阈值随机初始化，相同结构网络每次预测结果都不相同，因此算法优化效果有限。有学者提出一种用遗传算法同时优化 BP 神经网络结构和权值的算法，个体编码分为两部分，前面一部分表示网络结构，后面一部分表示权值，但是由于个体长度不相同，个体间无法进行交叉操作，因此该方法的可用性不高。

3.4.3　算法的局限性

遗传算法优化 BP 神经网络是对普通 BP 神经网络的一种优化方法，如果把 BP 神经网络看成是一个预测函数，遗传算法优化 BP 神经网络相当于优化预测函数中的参数，优化后 BP 神经网络的预测效果一般优于未优化的 BP 网络。但是该算法是有局限性的，它只能有限提高原有 BP 神经网络的预测精度，并不能把预测误差较大的 BP 神经网络优化为能够准确预测的 BP 神经网络。尤其对一些因为样本数量少、样本分布不均匀而造成神经网络预测误差大的问题，优化后的网络预测能力一般不能得到明显提高。

参考文献

[1] 吴仕勇. 基于数值计算方法的 BP 神经网络及遗传算法的优化研究[D]. 昆明:云南师范大学,2006.

[2] 李明. 基于遗传算法改进的 BP 神经网络的城市人居环境质量评价研究[D]. 沈阳:辽宁师范大学,2007.

[3] 王学会. 遗传算法和 BP 网络在发酵模型中的应用[D]. 天津:天津大学,2007.

[4] 李华. 基于一种改进遗传算法的神经网络[D]. 太原:太原理工大学,2007.

[5] 侯林波. 基于遗传神经网络算法的基坑工程优化反馈分析[D]. 大连:大连海事大学,2009.

[6] 吴建生. 基于遗传算法的 BP 神经网络气象预测建模[D]. 南宁:广西师范大学,2004.

[7] 黄继红. 基于改进 PSO 的 BP 网路的研究应应用[D]. 长沙:长沙理工大学,2008.

[8] 段侯峰. 基于遗传算法优化 BP 神经网络的变压器故障诊断[D]. 北京:北京交通大学,2008.

第 4 章　神经网络遗传算法函数极值寻优

——非线性函数极值寻优

4.1　案例背景

对于未知的非线性函数，仅通过函数的输入输出数据难以准确寻找函数极值。这类问题可以通过神经网络结合遗传算法求解，利用神经网络的非线性拟合能力和遗传算法的非线性寻优能力寻找函数极值。本章用神经网络遗传算法寻优如下非线性函数极值，该函数表达式为

$$y = x_1^2 + x_2^2 \tag{4-1}$$

函数的图形如图 4-1 所示。

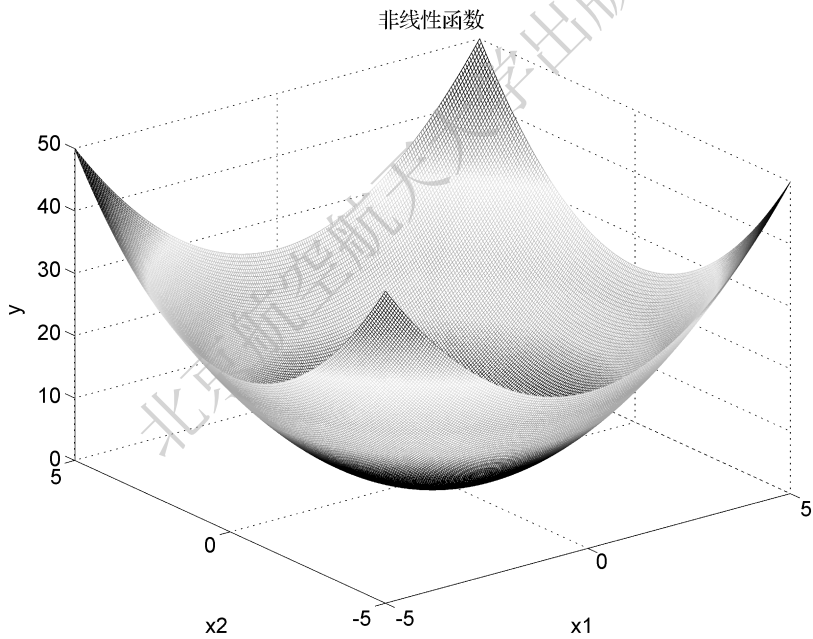

图 4-1　非线性函数图形

从函数方程和图形可以看出，该函数的全局最小值为 0，对应的坐标为(0,0)。虽然从函数方程和图形中很容易找出函数极值及极值对应坐标，但是在函数方程未知的情况下函数极值及极值对应坐标就很难找到。

4.2 模型建立

神经网络遗传算法函数极值寻优主要分为 BP 神经网络训练拟合和遗传算法极值寻优两步，算法流程如图 4－2 所示。

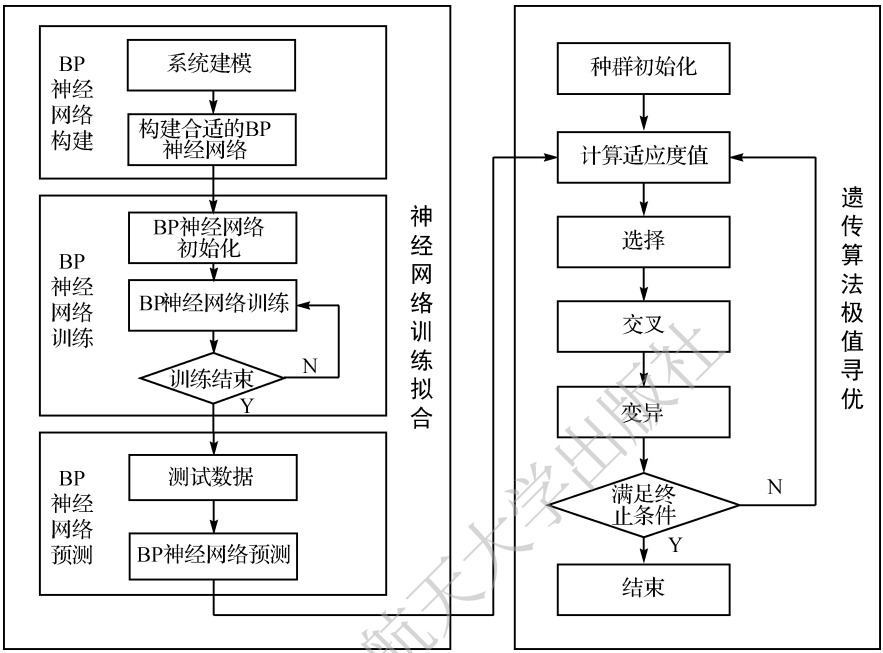

图 4－2　算法流程图

神经网络训练拟合根据寻优函数的特点构建合适的 BP 神经网络，用非线性函数的输入输出数据训练 BP 神经网络，训练后的 BP 神经网络就可以预测函数输出。遗传算法极值寻优把训练后的 BP 神经网络预测结果作为个体适应度值，通过选择、交叉和变异操作寻找函数的全局最优值及对应输入值。

对于本案例来说，根据非线性函数有 2 个输入参数、1 个输出参数，确定 BP 神经网络结构为 2—5—1。取函数的 4 000 组输入输出数据，从中随机选取 3 900 组数据训练网络，100 组数据测试网络性能，网络训练好后用于预测非线性函数输出。

遗传算法中个体采用实数编码，由于寻优函数只有 2 个输入参数，所以个体长度为 2。个体适应度值为 BP 神经网络预测值，适应度值越小，个体越优。选择算子、交叉算子和变异算子同第 3 章介绍各算子一致，交叉概率为 0.4，变异概率为 0.2。

4.3 编程实现

根据神经网络和遗传算法原理，在 MATLAB 中编程实现神经网络遗传算法非线性函数寻优。

4.3.1 BP 神经网络训练

用函数输入输出数据训练 BP 神经网络，使训练后的网络能够拟合非线性函数输出，保存训练好的网络用于计算个体适应度值。根据非线性函数方程随机得到该函数的 4 000 组输入输出数据，存储于 data 中，其中 input 为函数输入数据，output 为函数对应输出数据，从中随机抽取 3 900组训练数据训练网络，100 组测试数据测试网络拟合性能。最后保存训练好的网络。

```
%下载输入输出数据
load data input output

%从 1 到 4000 随机排序
k = rand(1,4000);
[m,n] = sort(k);

%找出训练数据和预测数据
input_train = input(n(1:3900),:)';
output_train = output(n(1:3900),:)';
input_test = input(n(3901:4000),:)';
output_test = output(n(3901:4000),:)';

%数据归一化
[inputn,inputps] = mapminmax(input_train);
[outputn,outputps] = mapminmax(output_train);

%构建 BP 神经网络
net = newff(inputn,outputn,5);

net.trainParam.epochs = 100;
net.trainParam.lr = 0.1;
net.trainParam.goal = 0.0000004;

%BP 神经网络训练
net = train(net,inputn,outputn);

%测试样本归一化
inputn_test = mapminmax('apply',input_test,inputps);

%BP 神经网络预测
an = sim(net,inputn_test);

%预测结果反归一化 BPoutput = mapminmax('reverse',an,outputps);

%网络存储
save data net inputps outputps
```

```
%网络预测图形
figure(1)
plot(BPoutput,':og')
hold on
plot(output_test,'- *');
legend('预测输出', '期望输出',fontsize',12)
title('BP网络预测输出','fontsize',12)
xlabel('样本','fontsize',12)
ylabel('输出','fontsize',12)
```

4.3.2 适应度函数

把训练好的BP神经网络预测输出作为个体适应度值。

```
function fitness = fun(x)
%计算个体适应度值
% x             input          个体
% fitness       output         个体适应度值

%神经网络下载
load data net inputps outputps

%输入数据归一化
inputn_test = mapminmax('apply',x,inputps);    %x为个体

%神经网络预测
an = sim(net,inputn_test);

%预测数据反归一化做为适应度值
fitness = mapminmax('reverse',an,outputps);
```

4.3.3 遗传算法主函数

遗传算法主函数MATLAB代码如下。

```
%清空环境变量
clc
clear

%遗传算法参数
maxgen = 100;                          %进化次数
sizepop = 20;                          %种群规模
pcross = 0.4;                          %交叉概率
pmutation = 0.2;                       %变异概率

lenchrom = [1 1];                      %每个变量长度
```

```
bound = [-5 5;-5 5];                    %变量边界
individuals = struct('fitness',zeros(1,sizepop),'chrom',[]);   %种群信息结构体
avgfitness = [];                        %种群每代平均适应度值
bestfitness = [];                       %种群每代最优适应度值
bestchrom = [];                         %最优个体
%初始化个体
for i = 1:sizepop
    %个体初始化
    individuals.chrom(i,:) = Code(lenchrom,bound);
    x = individuals.chrom(i,:);
    %个体适应度值
    individuals.fitness(i) = fun(x);
end
%寻找最优个体
[bestfitness bestindex] = min(individuals.fitness);
bestchrom = individuals.chrom(bestindex,:);
avgfitness = sum(individuals.fitness)/sizepop; %平均适应度值
%记录最优适应度和平均适应度
trace = [avgfitness bestfitness];
%迭代开始
for i = 1:maxgen
    %选择操作
    individuals = Select(individuals,sizepop);
    avgfitness = sum(individuals.fitness)/sizepop;
    %交叉操作
    individuals.chrom = Cross(pcross,lenchrom,individuals.chrom,sizepop,bound);
    %变异操作
    individuals.chrom = Mutation(pmutation,lenchrom,individuals.chrom,sizepop,[i maxgen],bound);
    %计算适应度值
    for j = 1:sizepop
        x = individuals.chrom(j,:);
        individuals.fitness(j) = fun(x);
    end

    %找最优和最差个体
    [newbestfitness,newbestindex] = min(individuals.fitness);
    [worestfitness,worestindex] = max(individuals.fitness);
    %更新最优个体
    if bestfitness>newbestfitness
        bestfitness = newbestfitness;
        bestchrom = individuals.chrom(newbestindex,:);
    end
    individuals.chrom(worestindex,:) = bestchrom;
    individuals.fitness(worestindex) = bestfitness;
```

```
    avgfitness = sum(individuals.fitness)/sizepop;

    %记录该代最优适应度和最差适应度
    trace = [trace;avgfitness bestfitness];
end

%遗传算法进化过程曲线
figure(1)
plot(trace(:,2))
title('适应度变化曲线','fontsize',12)
xlabel('进化次数','fontsize',12)
ylabel('适应度','fontsize',12)
```

4.3.4 结果分析

1. BP 神经网络拟合结果分析

本案例中个体的适应度值为 BP 神经网络预测值，因此 BP 神经网络预测精度对于最优位置的寻找具有非常重要的意义。由于寻优非线性函数有 2 个输入参数、1 个输出参数，所以构建的 BP 神经网络的结构为 2—5—1。共取非线性函数 4 000 组输入输出数据，从中随机选择 3 900 组数据训练 BP 神经网络，100 组数据作为测试数据测试 BP 神经网络拟合性能，BP 神经网络预测输出和期望输出对比如图 4－3 所示。

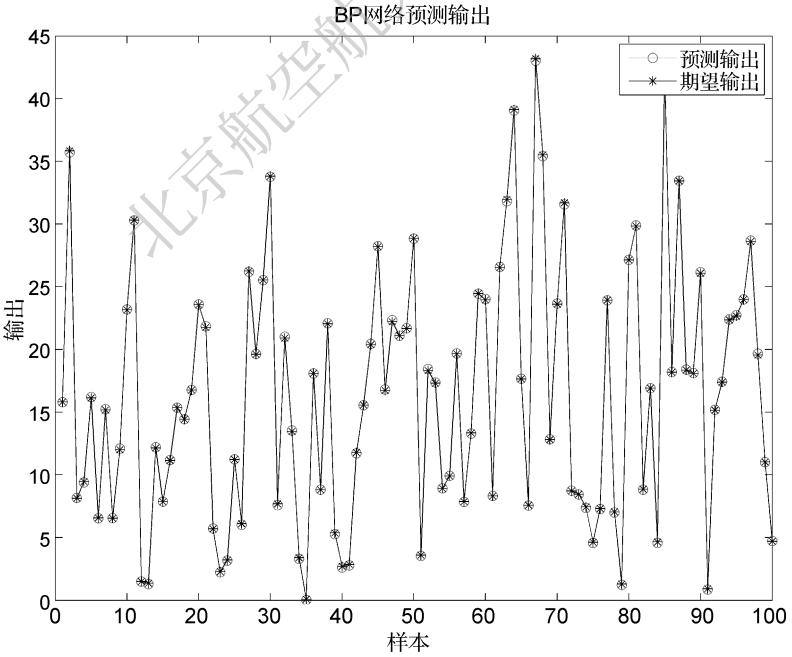

图 4－3　网络预测输出

从 BP 神经网络预测结果可以看出，BP 神经网络可以准确预测非线性函数输出，可以把网络预测输出近似看成函数实际输出。

2. 遗传算法寻优结果分析

BP 神经网络训练结束后，可以用遗传算法寻找该非线性函数的最小值，遗传算法的迭代次数是 100 次，种群规模是 20，交叉概率为 0.4，变异概率为 0.2，采用浮点数编码，个体长度为 2，优化过程中最优个体适应度值变化曲线如图 4－4 所示。

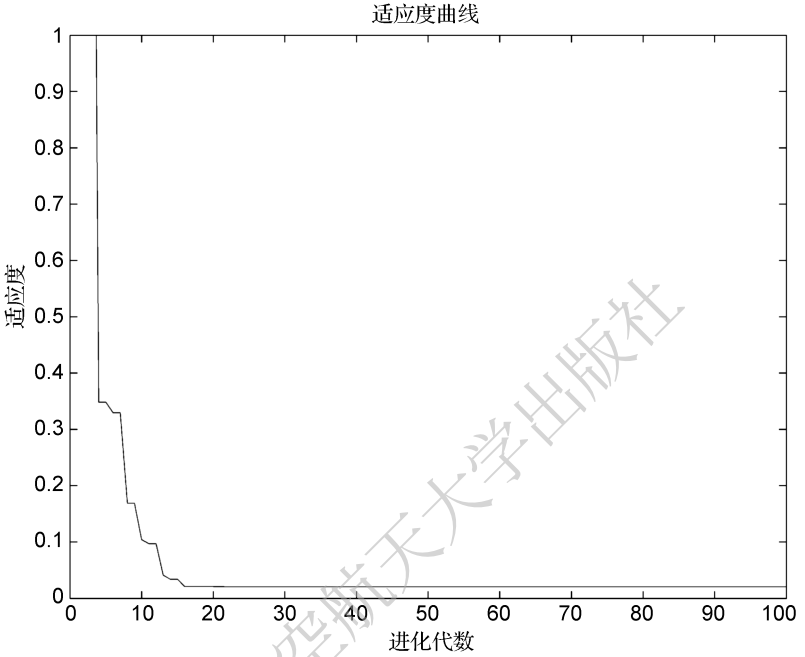

图 4－4　适应度变化曲线

遗传算法得到的最优个体适应度值为 0.020 6，最优个体为[0.000 3　－0.009 0]，最优个体适应度值同非线性函数实际最小值 0 和最小值对应坐标(0,0)非常接近，说明了该方法的有效性。

4.4　案例扩展

4.4.1　工程实例

本案例所使用的方法有比较重要的工程应用价值，比如对应某项试验来说，试验目的是得到最大试验结果对应下的试验条件。但是由于时间和经费限制，该试验只能进行有限次，可能单靠试验结果找不到最优的试验条件。这时可以在已知试验数据的基础上，通过本案例介绍的神经网络遗传算法寻找最优试验条件。思路为首先根据试验条件数和试验结果数确定 BP 神经网络结构，然后把试验条件作为输入数据，试验结果作为输出数据训练 BP 网络，训练后的网络就可以预测一定试验条件下的试验结果。然后把试验条件作为遗传算法中种群个体，把网络预测的试验结果作为个体适应度值，通过遗传算法推导最优试验结果及其对应试验条

件。已知实验数据如表 4.1 所列。

表 4.1　实验数据

实验条件				实验结果
添加物/kg	温度/(℃)	添加物/kg	时间/s	产量/kg
0	0	1 700	60	258
10	0	1 700	60	272
30	0	1 700	60	312
50	0	1 700	60	363
0	5	1 650	80	360
0	10	1 700	40	493
0	15	1 700	60	605
0	20	1 750	60	400
10	10	1 650	40	464
10	15	1 700	60	627
10	20	1 750	80	406
30	5	1 750	40	390
30	10	1 650	80	519
30	15	1 700	60	662
50	5	1 650	80	456
50	10	1 750	60	523
50	15	1 700	60	712
50	20	1 700	40	555

在试验中获得的最大实验结果为 712,对应的实验条件为[50　15　1 700　60]。在实验数据的基本上,采用神经网络遗传算法寻优。选择的 BP 网络结构为 4—10—1,遗传算法的迭代次数是 100 次,种群规模是 20,交叉概率 0.4,变异概率 0.2,采用浮点数编码,个体长度为 4,最后得到的最优实验结果为 745,对应的实验条件为[42.2　16.6　1 692.6　64.6],该结果可以为最优实验条件的选择提供参考。

4.4.2　预测精度探讨

BP 神经网络预测精度的好坏和寻优结果有着密切的关系。BP 神经网络预测越准确,寻优得到的最优值越接近实际最优值,这就需要在网络训练时采用尽可能多的训练样本。笔者曾经做过两个类似问题,一个是寻找 3 输入 4 输出系统的最大输出对应最优输入值,训练样本多达上万,神经网络预测效果非常好,最后得到预测最优值和真实最优值非常相近,误差在 10%以内。一个是寻找 3 输入 3 输出系统的最大输出对应输入值,训练样本只有三百多,神经网络预测的误差较大,最后寻优得到的最优值和真实最优值的误差在 20%以上。并且由于 BP 神经网络的拟合性能的局限性,并不是所有的系统都能够用 BP 神经网络精确表达,在方法使用上应该加以注意。

参考文献

[1] 邓虎.基于神经网络和遗传算法的凸轮轴数控磨削工艺参数优化[D].长沙:湖南大学,2008.

[2] 刘福国.双人工神经网络建模及约束条件下的遗传优化[J].动力工程,2007,27(3):357－361.

[3] L-异亮氨酸发酵培养基的遗传算法优化及发酵过程的神经网络建模[J].天津师范大学学报,2003,23(1):46－50.

[4] 方柏山.基于神经网络和遗传算法的木糖醇发酵培养基优化研究[J].生物工程学报,2000,16(5):648－650.

[5] L-机氨酸发酵培养基的神经网络建模与遗传算法优化[J].生物技术通讯,2005,16(2):156－158.

第 5 章　基于 BP_Adaboost 的强分类器设计

——公司财务预警建模

5.1　案例背景

5.1.1　BP_Adaboost 模型

Adaboost 算法的思想是合并多个“弱”分类器的输出以产生有效分类。其主要步骤为：首先给出弱学习算法和样本空间(x,y)，从样本空间中找出 m 组训练数据，每组训练数据的权重都是 $1/m$。然后用弱学习算法迭代运算 T 次，每次运算后都按照分类结果更新训练数据权重分布，对于分类失败的训练个体赋予较大权重，下一次迭代运算时更加关注这些训练个体。弱分类器通过反复迭代得到一个分类函数序列 $f_1,f_2,\cdots,f_T$，每个分类函数赋予一个权重，分类结果越好的函数，其对应权重越大。T 次迭代之后，最终强分类函数 F 由弱分类函数加权得到。**BP_Adaboost 模型即把 BP 神经网络作为弱分类器，反复训练 BP 神经网络预测样本输出，通过 Adaboost 算法得到多个 BP 神经网络弱分类器组成的强分类器。**

5.1.2　公司财务预警系统介绍

公司财务预警系统是为了防止公司财务系统运行偏离预期目标而建立的报警系统，具有针对性和预测性等特点。它通过公司的各项指标综合评价并预测公司财务状况、发展趋势和变化，为决策者科学决策提供智力支持。

财务危机预警指标体系中的指标可分为表内信息指标、盈利能力指标、偿还能力指标、成长能力指标、线性流量指标和表外信息指标六大指标，每项大指标又分为若干小指标，如盈利能力指标又可分为净资产收益率、总资产报酬率、每股收益、主营业务利润率和成本费用利润率等。在用于公司财务预警预测时，如果对所有指标都进行评价后综合，模型过于复杂，并且各指标间相关性较强，因此在模型建立前需要筛选指标。

指标筛选分为显著性分析和因子分析两步。显著性分析通过 T 检验方法分析 ST 公司和非 ST 公司，在财务指标中找出差别较大、能够明显区分两类公司的财务指标。因子分析在显著性分析基础上对筛选出来的指标计算主成分特征值，从中找出特征值大的指标作为公司危机预警方法的最终评价指标。最终找出成分费用利润率、资产营运能力、公司总资产、总资产增长率、流动比率、营业现金流量、审计意见类型、每股收益、存货周转率和资产负债率十项指标作为评价指标，该十项指标能够比较全面地反映出公司的财务状况。

5.2　模型建立

基于 BP_Adaboost 模型的公司财务预警算法流程如图 5－1 所示。

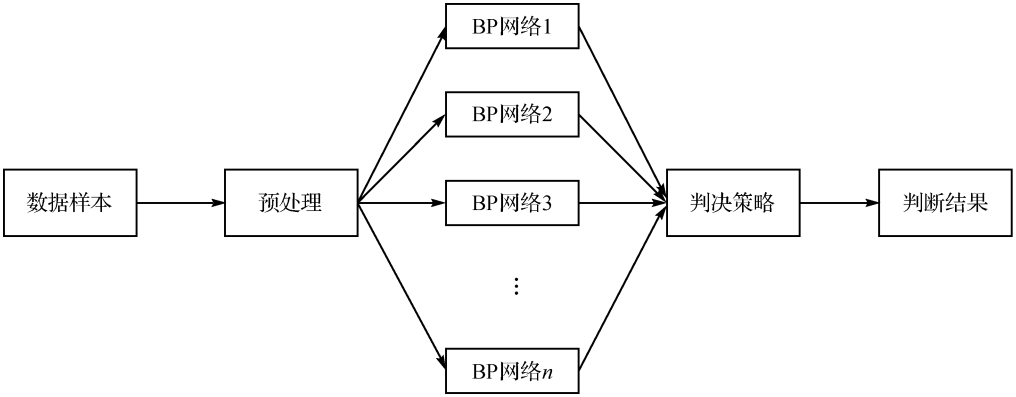

图 5-1　算法流程

算法步骤如下。

步骤 1:数据选择和网络初始化。从样本空间中随机选择 m 组训练数据,初始化测试数据的分布权值 $D_t(i)=1/m$,根据样本输入输出维数确定神经网络结构,初始 BP 神经网络权值和阈值。

步骤 2:弱分类器预测。训练第 t 个弱分类器时,用训练数据训练 BP 神经网络并且预测训练数据输出,得到预测序列 $g(t)$的预测误差和 e_t,误差和 e_t 的计算公式为

$$e_t = \sum_i D_i(i) \quad i = 1,2,\cdots,m(g(t) \neq y) \tag{5-1}$$

式中,$g(t)$为预测分类结果;y 为期望分类结果。

步骤 3:计算预测序列权重。根据预测序列 $g(t)$的预测误差 e_t 计算序列的权重 a_t,权重计算公式为

$$a_t = \frac{1}{2}\mathrm{In}\left(\frac{1-e_t}{e_t}\right) \tag{5-2}$$

步骤 4:测试数据权重调整。根据预测序列权重 a_t 调整下一轮训练样本的权重,调整公式为

$$D_{t+1}(i) = \frac{D_t(i)}{B_t} * \exp[-a_t y_i g_t(x_i)] \quad i = 1,2,\cdots,m \tag{5-3}$$

式中,B_t 是归一化因子,目的是在权重比例不变的情况下使分布权值和为 1。

步骤 5:强分类函数。训练 T 轮后得到 T 组弱分类函数 $f(g_t,a_t)$,由 T 组弱分类函数 $f(g_t,a_t)$组合得到了强分类函数 $h(x)$。

$$h(x) = \mathrm{sign}\left[\sum_{t=1}^{T} a_t \cdot f(g_t,a_t)\right] \tag{5-4}$$

对于本案例来说,共有 1 350 组公司财务状况数据,每组数据的输入为 10 维,代表上述的 10 个指标,输出为 1 维,代表公司财务状况,为 1 时表示财务状况良好,为−1 时表示财务状况出现问题。从中随机选取 1 000 组数据作为训练数据,350 组数据作为测试数据。根据数据维数,采用的 BP 神经网络结构为 10—6—1,共训练生成 10 个 BP 神经网络弱分类器,最后用 10 个弱分类器组成强分类器对公司财务状况进行分类。

5.3 编程实现

根据 Adaboost 和 BP 神经网络原理，编程实现基于 BP_Adaboost 算法的公司财务预警建模。

5.3.1 数据集选择

从样本空间中选择训练样本，测试样本，并对测试样本分配权重，其中训练数据和测试数据存储在 data 文件中，input_train，output_train 为训练输入输出数据，input_test，output_test 为预测输入输出数据。

```
%清空环境变量
clc
clear

%下载数据
load data input_train output_train input_test output_test

%测试样本权重
[mm,nn] = size(input_train);
D(1,:) = ones(1,nn)/nn;
```

5.3.2 弱分类器学习分类

把 BP 神经网络看作弱分类器，经过训练后分类训练样本，并且根据训练样本分类结果调整训练样本权重值，最终得出一系列弱分类器及其权重。为了体现出强分类器分类效果，本例降低了 BP 神经网络训练次数以降低弱分类器分类能力。

```
K = 10;  %若分类器数量
for i = 1:K

    %训练样本归一化
    [inputn,inputps] = mapminmax(input_train);
    [outputn,outputps] = mapminmax(output_train);
    error(i) = 0;

    %BP 神经网络构建
    net = newff(inputn,outputn,6);
    net.trainParam.epochs = 4;
    net.trainParam.lr = 0.1;
    net.trainParam.goal = 0.00004;

    %BP 神经网络训练
    net = train(net,inputn,outputn);
```

```
    %训练数据预测
    an1 = sim(net,inputn);
    test_simu1(i,:) = mapminmax('reverse',an1,outputps);

    %测试数据预测
    inputn_test = mapminmax('apply',input_test,inputps);
    an = sim(net,inputn_test);
    test_simu(i,:) = mapminmax('reverse',an,outputps);

    %统计输出结果
    kk1 = find(test_simu1(i,:)>0);
    kk2 = find(test_simu1(i,:)<0);

    aa(kk1) = 1;
    aa(kk2) = -1;

    %统计错误样本数
    for j = 1:nn
        if aa(j)~ = output_train(j);
            error(i) = error(i) + D(i,j);
        end
    end

    %若分类器 i 权重
    at(i) = 0.5 * log((1 - error(i))/error(i));

    %更新 D 值
   for j = 1:nn
        D(i + 1,j) = D(i,j) * exp( - at(i) * aa(j) * test_simu1(i,j));
    end

    %D 值归一化
    Dsum = sum(D(i + 1,:));
    D(i + 1,:) = D(i + 1,:)/Dsum;
end
```

5.3.3 强分类器分类和结果统计

由 10 组弱分类器 BP 网络组成强分类器对分析样本进行分类,并统计分类误差。

```
%强分类器分类结果
output = sign(at * test_simu);
%统计强分类器每类分类错误个数
kkk1 = 0;
kkk2 = 0;
for j = 1:350
```

```
        if output(j) == 1
            if output(j)~= output_test(j)
                kkk1 = kkk1 + 1;
            end
        end
        if output(j) == -1
            if output(j)~ = output_test(j)
                kkk2 = kkk2 + 1;
            end
        end
    end

    %统计弱分类器每类分类误差个数
    for i = 1:K
        error1(i) = 0;
        kk1 = find(test_simu(i,:)>0);
        kk2 = find(test_simu(i,:)<0);

        aa(kk1) = 1;
        aa(kk2) = -1;

        for j = 1:350
            if aa(j)~ = output_test(j);
                error1(i) = error1(i) + 1;
            end
        end
    end
```


5.3.4 结果分析

分析样本共有 350 组数据，采用 10 个 BP 弱分类器组成的强分类器分类公司财务运行状况，分类误差统计如表 5.1 所列。

表 5.1 分类误差统计

强分类器分类误差率	弱分类器分类平均误差率
4.00%	6.37%

从表 5.1 可以看出，强分类器分类误差率低于弱分类器分类误差率，表明 BP_Adaboost 分类算法取得了良好的效果。

5.4 案例扩展

Adaboost 方法不仅可以用于设计强分类器，还可以用于设计强预测器。强预测器设计思路与强分类器设计类似，都是先赋予测试样本权重，然后根据弱预测器预测结果调整测试样本

权重并确定弱预测器权重，最后把弱预测器序列作为强预测器。不同的是在强分类器中增加预测错类别样本的权重，在强预测器中增加预测误差超过阈值的样本权重。采用 BP_Adaboost 强预测器预测第 2 章中非线性函数输出，函数形式为 $y=x_1^2+x_2^2$。

BP 神经网络参数设置见第 2 章，MATLAB 程序如下。

5.4.1 数据集选择

从样本空间中选择训练样本，测试样本，并对测试样本分配权重。函数 $y=x_1^2+x_2^2$ 的输入输出数据存储在 data1.mat 文件中，其中 input 为函数输入数据，output 为函数输出数据，从中随机选择 1 900 组数据作为训练数据，100 组数据作为测试数据。

```
%清空环境变量
clc
clear

%下载数据
load data1 input output

%从中随机选择 1900 组训练数据和 100 组测试数据
k = rand(1,2000);
[m,n] = sort(k);

%训练样本
input_train = input(n(1:1900),:)';
output_train = output(n(1:1900),:)';

%测试样本
input_test = input(n(1901:2000),:)';
output_test = output(n(1901:2000),:)';
%样本权重
[mm,nn] = size(input_train);
D(1,:) = ones(1,nn)/nn;
```


5.4.2 弱预测器学习预测

把 BP 神经网络看作弱预测器，经过训练后预测测试样本输出，并且根据预测结果调整样本测试样本权重值，最终得出一系列弱预测器及其权重。这里把预测误差超过 0.1 的测试样本作为应该加强学习的样本。

```
K = 10;
%循环开始
for i = 1:K

    %弱预测器训练
    net = newff(inputn,outputn,5);
    net.trainParam.epochs = 20;
```